CONNEXIONS™

The Interoperability Report

March 1989 | *Special Issue: Network Management* | Volume 3, No. 3

ConneXions—
The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published by Advanced Computing Environments, 480 San Antonio Road, Suite 100, Mountain View, CA 94040, USA. Phone: 415-941-3399.

ISSN 0894-5926

From the Editor

Network Management has become an important topic for research and development over the last couple of years. So great has been the interest in this arena, that a number of competing and incompatible systems have been proposed. In the Internet community we had, last year, no less than three systems: *The High-level Entity Management System* (HEMS), *The Simple Network Management Protocol* (SNMP), and *CMIP over TCP* (CMOT). This situation was clearly not beneficial to either the vendors or users of TCP/IP products, and it was because of this that a meeting was called in February 1988 of all the affected parties to determine the future of network management in the Internet. The resolution, as outlined in RFC 1052, calls for SNMP as the short-term Internet standard and CMOT as the future, ISO-based, network management system.

The February meeting which led to RFC 1052 was chaired by Vint Cerf, and it is therefore appropriate that we begin this issue with his thoughts on network management in the Internet.

Marshall Rose and Keith McCloghrie describe the *Management Information Base* (MIB) and the Structure of *Management Information* (SMI) which were developed in a fashion that make them applicable to both the SNMP and CMOT architectures.

The various Internet network management working groups have found *Case Diagrams* a particularly useful tool for designing management information bases. Jeff Case and Craig Partridge explain the method in an article on page 10.

The CMOT architecture is outlined by Amatzia Ben-Artzi on page 14, and in a companion article George Marshall gives a brief description of the NetMan demonstration at INTEROP™ 88.

The SNMP authors, Jeff Case, James Davin, Mark Fedor, and Marty Schoffstall introduce the system starting on page 22.

Early implementation and experimentation is an important step on the road to full OSI network management. The University College London (UCL) has participated in the EEC-funded INCA project and implemented an OSI network management system. Graham Knight of UCL outlines this project on page 27.

Finally, Unni Warrier of Unisys gives an overview of a high-level *Network Management Language* which can support a variety of underlying protocols.

Network Management Comes of Age in the Internet

History

In the earliest days of the Internet research project, most of the attention was placed on the matter of end-to-end protocols that could provide reliable channels across a variety of packet networks. Gateway functions were defined largely in terms of supporting the end-to-end protocol requirements. There was only a single implementor of gateways, Bolt Beranek and Newman, and BBN used techniques and methods similar to those employed in the Arpanet and its components (packet switches and terminal access controllers) to configure and manage the Internet gateway subsystem.

Once the TCP/IP protocol suite was adopted as a DoD standard, a number of vendors began to develop commercial products based on these protocols. Moreover, a number of gateway vendors emerged, each of whose products offered different features. The vendors recognized the need for network management tools to help their customers operate the private internets built from vendor products.

Two protocols

By February, 1988, it was widely understood by both the research and vendor community, that Internet-wide network management tools were an essential requirement to cope with the rapid growth and scaling of the system. At that time, several possible protocols for supporting network management mechanisms were under active development. A year later, there are essentially two basic protocols offered: *Simple Network Management Protocol* (SNMP) and *Common Management Information Protocol over TCP* (CMOT). SNMP is an IAB-approved protocol for current use. CMOT is maturing as an alternative, longer-term protocol which is intended to be compatible with international standards efforts in this area.

The importance of gaining experience with the application of these protocols cannot be overstressed. Moreover, it is critical that we be able to develop network management applications which can operate on either of the two protocols so as to retain the investment in applications using SNMP when CMOT-based software becomes available. The Internet Engineering Task Force is organizing work in this area to focus attention on common application interfaces to the protocols and to develop specific *Management Information Base* variables for layer-specific internet subsystem management. This work is important in several respects, not the least of which is the need to fashion the resulting software so that it can be applied to several protocol suites. The heterogeneous nature of networking is likely to persist and the management tools we build today must be able to serve us across a range of vendor products and protocols.

Common tools

As a final note, I want to stress the critical nature of multi-vendor implementation of common network management tools. The user community lives in a multi-vendor environment. This results both from still-useful installed base, to which is added new equipment with new features and because users are becoming far more sophisticated in their choice of equipment. They often deliberately choose multi-vendor systems to optimize certain desired functionality. A consequence is that users now need and insist on common tools with which to operate and manage the resulting complex.

The challenge lies before the vendor and the research community to develop, test and deploy satisfactory solutions to these problems.

—Vint Cerf, Corporation for National Research Initiatives.

Network Management of TCP/IP-based internets

by **Keith McCloghrie and Marshall T. Rose**
The Wollongong Group, Inc.

Introduction

As networks grow larger and more complex, ad hoc methods for managing them become unworkable. Initially, the use of TCP/IP was limited primarily to simple network topologies, and there was little perceived need for integrated network management. Today however, given the unprecedented success of TCP/IP as the de facto standard for computer networking, it is not surprising that network management is essential for maintaining successful day-to-day operations. As a result, there has been a tremendous interest in network management of TCP/IP-based internets.

IAB recommendations

On the 29th of February, 1988, an ad hoc committee met at the request of the Internet Activities Board (the body which oversees the technical development of TCP/IP). The committee's charter was to examine the current state of network management technology and provide recommendations to the IAB as to what steps should be taken to develop an Internet standard for network management. After consideration of the committee's deliberations, the IAB issued RFC 1052 detailing their strategy. A two-pronged approach was adopted:

- In the short-term, an existing management protocol, the *Simple Gateway Monitoring Protocol* (SGMP) would be modified to reflect the experience gained from its use in operational networks and would be called the *Simple Network Management Protocol* (SNMP); and,

- In the long-term, use of the emerging OSI network management protocol, the *Common Management Information Protocol* (CMIP) would be investigated.

The disadvantage of a two-pronged approach is, of course, that a transition is needed from the short-term to the long-term solution. In order to provide an orderly transition between the technologies, it is essential that a common framework for management of TCP/IP-based internets be developed, and that both solutions work within that framework.

Working Groups

As a result of the IAB recommendations, 3 working groups of the Internet Engineering Task Force (IETF) of the IAB were chartered:

- The *Management Information Base* (MIB) working group was formed to develop the common framework for TCP/IP network management. This group is chaired by Craig Partridge of BBN Systems and Technologies Corporation.

- The *SNMP Extensions* working group was formed to make minor extensions to the SGMP based on operational experience and to achieve conformance with the results of the MIB working group. This working group met its charter by producing RFC 1067, and then promptly disbanded. The SNMP Extensions working group was chaired by Marshall T. Rose.

continued on next page

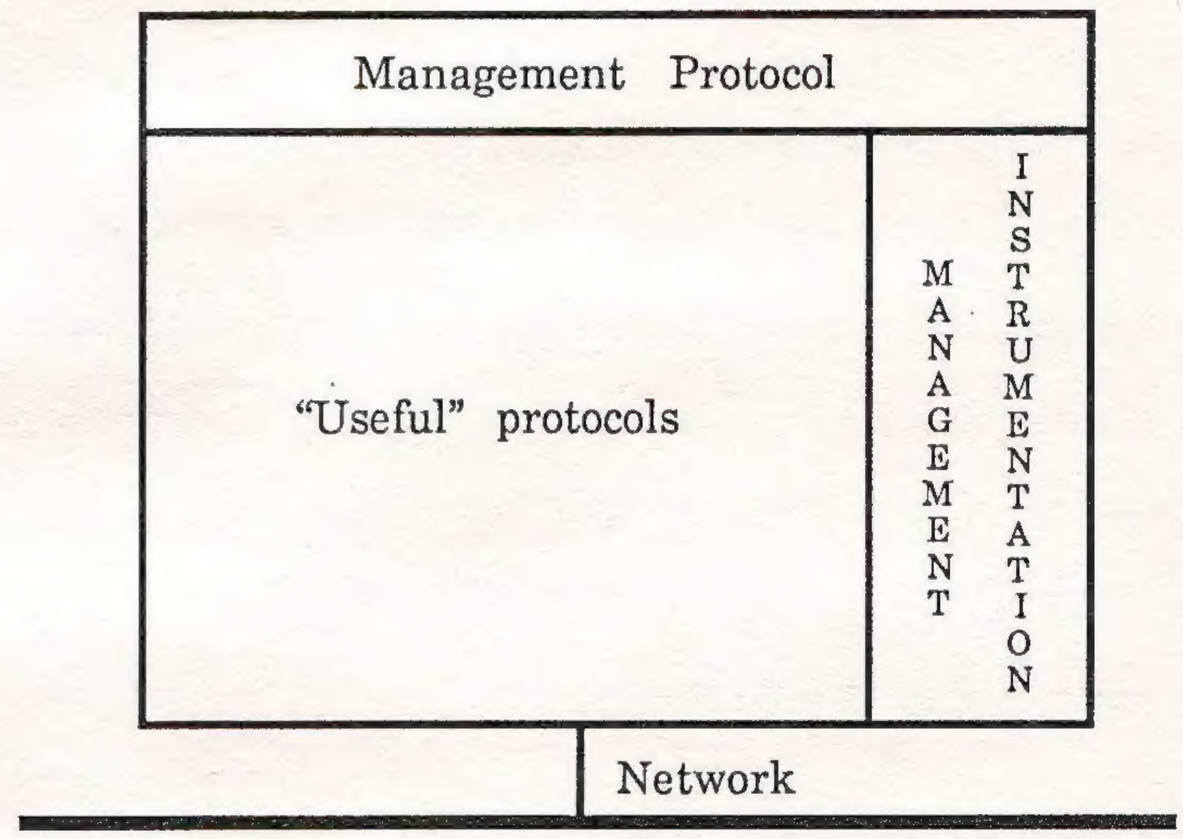
Network Management *(continued)*

- The *NetMan* working group was directed to continue its exploration of the OSI network management framework and develop CMOT—CMIP over TCP. This working group, which is nearing completion of its first phase, is chaired by Lee LaBarre of the MITRE Corporation.

The purpose of this article is to describe the results of the first working group. Companion articles in this gala issue of *ConneXions* will describe the Simple Network Management Protocol (SNMP) and CMOT.

The task

The MIB working group had a quite difficult task: in addition to enumerating the objects that could be managed, these objects had to be defined in a way which was independent of the actual management protocol in use. This protocol-independence is critical if more than one management protocol is ever to be used. To understand why this is so, consider the high-level architecture of a managed node, such as a gateway, as shown here:



A managed node conceptually consists of three parts:

- "*useful*" protocols, which do "real" work;
- a *management protocol*, which permits the monitoring and control of the "useful" protocols; and,
- *management instrumentation*, which interacts with the local implementation of the "useful" protocols in order to achieve monitoring and control.

Independence

In order to do network management, we think of the "useful" protocols as containing *managed objects*, such as routing tables, interface information, Telnet connections, and so on. In an implementation, the management instrumentation provides these managed object abstractions to the management protocol. Thus, to the extent that these managed objects are defined in a fashion that is independent of the management protocol, the management instrumentation is also independent of the management protocol. This means that one could conceivably exchange the management protocol used on a node without changing either the "useful" protocols or the management instrumentation.

In any well-designed system, the cost of implementing and maintaining the "useful" protocols should be at least an order of magnitude greater than the associated costs of the management instrumentation and protocols. Thus, if the objects are defined in a management protocol-independent fashion, a significant savings may be realized.

Another benefit of the common framework is obtained in management stations. When the transition from the SNMP to the CMOT occurs, there will undoubtedly be a mixture of managed nodes using either the SNMP or the CMOT. By having a common framework, the network operator can be presented with a common interface for managing the network, independent of what actual protocols are used to manage individual nodes.

SMI and MIB

To achieve this independence, the working group divided its task into two parts. The first part was to develop a *Structure of Management Information* or SMI. The SMI defines the rules for how managed objects are described and how management protocols may access these objects. The second part was to develop a *Management Information Base* or MIB. This is the collection of objects that can be accessed via a network management protocol.

For the remainder of this article, we now consider the SMI and MIB that was originally ratified by the MIB working group. It should be noted that the MIB working group meets regularly to deliberate on both the SMI and the MIB. Although the SMI is stable, the MIB is expected to grow considerably over the next few years.

Philosophy

The SMI, a draft standard for the Internet community, defined in RFC 1065, is largely an administrative document: it says "how to do things", but not actually "what to do." It is the MIB and the management protocol, which are responsible for the "what to do" part, and the SMI is used strictly to provide a well-defined interface between them. The SMI philosophy is to foster:

- *Simplicity*, because general understanding of network management is (so far) very limited; by taking a simple approach, future extensions will be less constrained. In addition, there is hope that today's systems will be more workable.

- *Extensibility*, because there are many possible approaches which might be followed; by emphasizing extensibility, a larger number of future approaches might be attempted. In addition, since today's systems will be around for a while, it is essential to provide an easy way for the new to work with the old.

ASN.1

The *object information model* used in the SMI is simple. Managed objects are defined using a data description language called ASN.1 (the OSI Abstract Syntax Notation One). ASN.1 is useful for describing structures in a machine-independent fashion. Additionally, ASN.1 definitions can be written which convey to the human reader the semantics of the objects they define.

The SMI provides an ASN.1 macro, `OBJECT-TYPE`, that the MIB uses when managed objects are defined. An example is given on the next page.

continued on next page

Network Management *(continued)*

```
sysUpTime
  OBJECT-TYPE
  SYNTAX  TimeTicks
  ACCESS  read-only
  STATUS  mandatory
  ::= { system 3 }
```

This simple definition captures the following semantics:

- A managed object called `sysUpTime` is defined;
- This object contains `TimeTicks` information (this datatype is also defined in the SMI—it is a measurement of time in hundredths of a second);
- This object is read-only, so that management protocols may not attempt to modify this object;
- This object is mandatory, so that all managed nodes (gateways, hosts, etc.) must provide this object; and,
- When a management protocol accesses this object, it uses the name `{ system 3 }`

This is all intuitive except for the last part.

Object Indentifiers

All managed objects must have a "handle" which can be used by a management protocol when it wishes to identify that object. The SMI uses the ASN.1 OBJECT IDENTIFIER notion for this purpose. An OBJECT IDENTIFIER is an administratively assigned name, just like a telephone number.

For example, an international number consists of a country code followed by a telephone number in that country, such as +1–415–941–3399. The first number, the country code, is defined by an International body. The second number, the telephone number, is defined by that country. Further administration may also be present. For example, in the North American Dialing Plan, each telephone number consists of an area code (e.g., 415), a central office number in that area code (e.g., 941), and then a station number attached to that central office (e.g., 3399). At each stage, a new administrative entity might be introduced to decide how numbers are allocated.

The same is true of OBJECT IDENTIFIER. The top-level numbers are allocated by the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC). In turn, the ISO/IEC has allocated some of the numbers to various national standards bodies. In turn, one of these bodies allocated a number to the U.S. Department of Defense. Finally, one of these numbers, 1.3.6.1.2, was allocated for TCP/IP-based network management. The SMI takes it from here, by saying how numbers that are subordinate to 1.3.6.1.2 are used. The name `{ system 3 }` is such a number which identifies the object as occurring in the current Internet MIB, in the system group, as the third defined object.

In addition to the Internet standard MIB, the SMI also allocates name-space for experimental objects and enterprise-specific objects. For the latter, vendors and organizations can apply for their own number and receive the delegated authority for defining their own managed objects.

Simplifications

One of the "simple" aspects of the SMI was to restrict the kind of data that could be used in a managed object. Many powerful and complex structures can be defined using ASN.1. These same structures are also slow to encode when transmitted on the network; they are also hard to decode when received from the network. In order to make network management available to the largest range of TCP/IP-based products, it was decided that the ASN.1 used to represent the "syntax" of a managed object should be limited. The SMI defines the rules which mandate this.

The SMI does not define access mechanisms used by the management protocols. For example, the SMI permits aggregate types, such as tables, to be defined. The SNMP does not support the retrieval of aggregate types, so it must access each object type within a table separately. In contrast, CMOT does support retrievals of more complex objects.

Version management

The final topic concerning the SMI deals with how the MIB evolves from one version to the next. When a version of the Internet standard MIB is produced, all of the objects it defines share a common naming structure of the form

```
prefix.version.tail
```

where "prefix" is 1.3.6.1.2 and "version" is the number of the MIB specification. Further, if an object is defined in more than one version of the MIB, then the same tail is used. The new version of the MIB is free to add more information to the object (e.g., a new column in a table), but may not remove information. Finally, if a version of the MIB marks an object as obsolete, the associated tail is never re-used. These rules give an important regularity to the way objects are named.

Suppose a network management station supporting version 7 of the MIB is managing a gateway supporting version 1 of the MIB. Obviously, it is impossible for the gateway to know about versions 2—7 of the MIB; and, while it is possible for the station to know about versions 1—6, this is impractical. With the naming regularity imposed by the SMI, the following scenario is possible:

1. The station sends a request to the gateway asking for the value of a variable in the version 7 MIB.

2. The gateway knows it does not support this object directly. However, it checks the prefix used by the station and sees that it matches the prefix used by its MIB.

3. The gateway then skips over the version number and sees if it has an object with the same tail.

continued on next page

Network Management *(continued)*

4. If not, then the gateway is sure than no useful information can be provided to the station. Presumably, such an object would have been first defined in a MIB later than the one supported by the gateway.

5. If the gateway has an object with the same tail, then it can simply return that value.

6. The station looks at the value it gets back. If it exactly matches the ASN.1 syntax defined in the version of the MIB it supports, then there is no problem.

7. If not, the station knows that it has been given a subset of the information it requested, and can proceed accordingly.

Extensibility

It should be understood that the extensibility issue is an important one: a gateway being deployed today with version 1 of the MIB may be in the field for a considerable amount of time. It is simply not practical to keep all the managed nodes and management stations in an internet current with the latest MIB versions. Without an extensibility mechanism, interoperability would be severely hindered.

The MIB, which is also a draft standard for the Internet community, is defined in RFC 1066. The Management Information Base is largely a technical document: for each layer in the TCP/IP protocol suite, the MIB defines the objects which may be managed. The first version of the MIB was defined to capture the most essential aspects of the fundamental protocols. As experience is gained, future versions of the MIB will "fill in the gaps." The MIB itself is divided into several groups of variables:

Group	Objects for
system	the managed node itself
interfaces	network attachments
at	IP address translation
ip	the Internet Protocol
icmp	the Internet Control Message Protocol
tcp	the Transmission Control Protocol
udp	the User Datagram Protocol
egp	the Exterior Gateway Protocol, used by nodes directly attached to the Internet backbone

Each managed node does not have to support all of these groups. Rather, it need support only those groups which are appropriate (e.g., hosts needn't support the EGP group). However, if a group is appropriate, then all objects in that group must be supported.

Few objects

There were extensive criteria for deciding if an object should be defined in the MIB. Perhaps the two most important were that an object be needed to configure or diagnose a node, and was previously known to be useful for management. Further, the MIB working group had as a goal to produce a first version of the MIB with no more than 100 objects. If this could be accomplished, then it would be easier for vendors to fully implement the MIB. (By the time the dust had settled, 116 objects were defined.)

Status

RFCs 1065 and 1066 were issued in August of 1988. The following month, the SMI and MIB were demonstrated (along with the SNMP) at INTEROP™88. Since that time, several vendors have been shipping network management software based on these draft Internet standards. The biggest challenge of both the SMI and the MIB remain: when the CMOT is ready for experimentation, will the SMI have achieved the right level of management protocol-independence; and, at what rate will new versions of the MIB be produced to meet the ever expanding requirements of network administrators and operators?

KEITH McCLOGHRIE is a Director of Software Engineering at The Wollongong Group, Inc., where his responsibilities include the development of network management as well as TCP/IP products for a variety of computer systems. He was co-author of RFC 1065 ("Structure and Identification of Management Information for TCP/IP-based internets") and RFC 1066 ("Management Information Base for Network Management of TCP/IP-based internets"). He has also participated in the IETF's SNMP Extensions Working Group and the Netman (CMOT) Working Group. McCloghrie received his B.Sc. in Mathematics from the University of Manchester in 1969.

MARSHALL T. ROSE is a Principal Software Engineer at The Wollongong Group, Inc., where he works on OSI protocols and transition strategies. He was co-author of RFC 1065 ("Structure and Identification of Management Information for TCP/IP-based internets") and RFC 1066 ("Management Information Base for Network Management of TCP/IP-based internets"). He also chaired the SNMP Extensions Working Group of the IETF which produced RFC 1067 ("A Simple Network Management Protocol"). These three documents form the short-term network management solution for TCP/IP-based networks. Rose received the Ph.D. degree in Information and Computer Science from the University of California, Irvine, in 1984.

Bibliography

For more information on the systems described in this issue of *ConneXions* we recommend the following documents:

Cerf, V.G.,"IAB recommendations for the development of Internet network management standards," RFC 1052.

McCloghrie, K. & Rose, M., "Structure and Identification of Management Information for TCP/IP-based internets," RFC 1065.

McCloghrie, K. & Rose, M., "Management Information Base for Network Management of TCP/IP-based internets," RFC 1066.

Case, J., Fedor, M., Schoffstall, M.L., Davin, J., "Simple Network Management Protocol," RFC 1067.

Defense Advanced Research Projects Agency, Internet Activities Board (IAB), "IAB Official Protocol Standards," RFC 1083.

Rose, M. "ISO presentation services on top of TCP/IP-based internets," RFC 1085.

Klerer, S., "OSI Management: An Overview," *IEEE Network*, Vol. 2, No. 2, March 1988.

"Common Management Information Services (CMIS)," ISO 9595/2.

"Common Management Information Protocol (CMIP)," ISO 9596/2.

Case Diagrams: A tool for diagraming management information bases

by Jeffrey Case, University of Tennessee at Knoxville
and Craig Partridge, BBN Systems and Technologies Corp.

Introduction

In the spring and early summer of 1988, the *MIB Working Group* of the Internet Engineering Task Force (IETF), chaired by Craig Partridge, was charged with developing a *Management Information Base* (MIB) for the TCP/IP protocol suite [1,2]. A MIB is a collection of management objects maintained on a network entity that may be remotely manipulated to achieve remote network management. During the meetings of the working group, the members often struggled to develop clear definitions of the objects we wanted to place in the MIB. At one of the meetings, Jeff Case suggested that the group try to diagram the management objects for each protocol layer. The resulting pictures, promptly dubbed "Case diagrams," proved invaluable for clarifying difficult definitions and for refining our decisions about what objects should be included in the MIB.

Although Case diagrams are an *ad hoc* solution and have some limitations, we believe that there will be a need in the future for a visual representation of MIB objects. The authors hope is that this article will help encourage additional research.

Why diagram a MIB?

The saying is "a picture is worth a thousand words." Our experience in the MIB working group suggests that pictures did indeed save us much wasted verbiage and time. (It is the chairman's opinion that Case diagrams allowed the MIB Working Group to produce a MIB definition in less than half the time it would have taken to define an inferior MIB without Case diagrams).

One of the goals of the MIB working group was to avoid redundant object definitions while ensuring that key information was indeed captured in at least one object. The reasons for wanting all key information are, we expect, clear. The desire to avoid redundant definitions stemmed from a concern that we might hurt network performance if protocol implementations contained excessive instrumentation. (Or, to put the problem more bluntly, the fewer counters you have to increment, the less time it takes to get the packet from an application out onto the network.)

A key problem was making sure that every protocol data unit (PDU) received at a layer was accounted for. The working group felt it was very important to know the fate of each PDU; was it sent, lost, in improper format? Unfortunately, it is very difficult to read a list of object definitions and be sure that all data paths are covered. You really need a diagram that shows all the data flows and how they are counted. Showing the data flows, in essence, is what Case diagrams are designed to do. (Indeed, the problem of tracking flows reminded Case of Kirchoff diagrams for the flow of current in electrical circuits).

Once the working group started to use Case diagrams, the process of developing definitions became much easier. The group was rapidly able to show that its previous attempts at comprehensive definitions were incorrect (some data flows were not covered), and within a few hours was able to develop a better set of object definitions.

Using a Case Diagram

The primary use of Case diagrams is to illustrate the relative positions of counters, objects which keep running totals, on input or output streams within a particular protocol layer. Consider, for example, inbound datagrams in the IP layer. Datagrams are received from the network access or interface layer below IP, are processed in the IP layer, and then those datagrams that are valid and complete (i.e., not fragmented) are passed up to the transport layer. For network management purposes, we would like to trace the various paths that datagrams may take through the IP layer, and also to account for all datagrams that may be discarded at the IP layer due to errors or lack of resources such as memory buffers.

The Case diagram for inbound IP datagrams is shown in Figure 1. The inbound datagrams are viewed as a vector from the Interface layer through to the Transport layer. Counters are indicated by smaller vectors that remove from, add to, or simply count datagrams on the vector. Position within the diagram is significant. Thus we can observe from the diagram that we expect all implementations to count the number of IP datagrams received (*ipInReceives*) before they deduct the counts for IP datagrams with errors in headers or bad addresses (*ipInHdrErrors* and *ipInAddrErrors* respectively). Further, we can observe that the IP fragment reassembly module removes fragments from the data path (*ipReasmReqds*), attempts to reassemble the fragments, and inserts correctly reassembled datagrams (*ipReasmOKs*) back into the data path. Fragments that could not be reassembled are also recorded (in *ipReasmFails*).

Case diagrams can also show the position of counters which count only selected datagrams. For example, in Figure 1, if we wished to keep track of the different types of destination addresses in the IP datagrams we received (for example, separately counting unicast, multicast and broadcast addressed datagrams) we could place these counters on the input stream.

Limitations of Case Diagrams

Case diagrams have their limitations. Currently, they can only be used to identify and locate counters. Objects that indicate state (for example, the status of a TCP connection, or the UP/DOWN status of an interface) cannot be represented in a Case diagram.

Another, more subtle problem is that Case diagrams assume that there is an ordering to counters; that one counter will always be adjusted before another. This is not always true. For example, in the IP layer diagram in Figure 1, the count of dropped IP datagrams, *ipInDiscards* may be incremented anywhere in the layer. (The IP layer can discard datagrams for lack of resources to process it; a shortage of resources may occur at any time before the datagram is passed to the transport layer).

Clearly Case diagrams could be enhanced to accommodate this additional information. However, we are concerned that enhancing the simple vector diagrams may make them too complex to interpret easily, and suspect that a more structured format may be required.

One last point to make about Case diagrams is that they are not a complete specification of a MIB and are not intended to be. While diagrams can display a considerable amount of information such as relative positions in the information flow, and object types (which are not currently kept in Case diagrams but could easily be added), they cannot fully define an object.

continued on next page

Case Diagrams *(continued)*

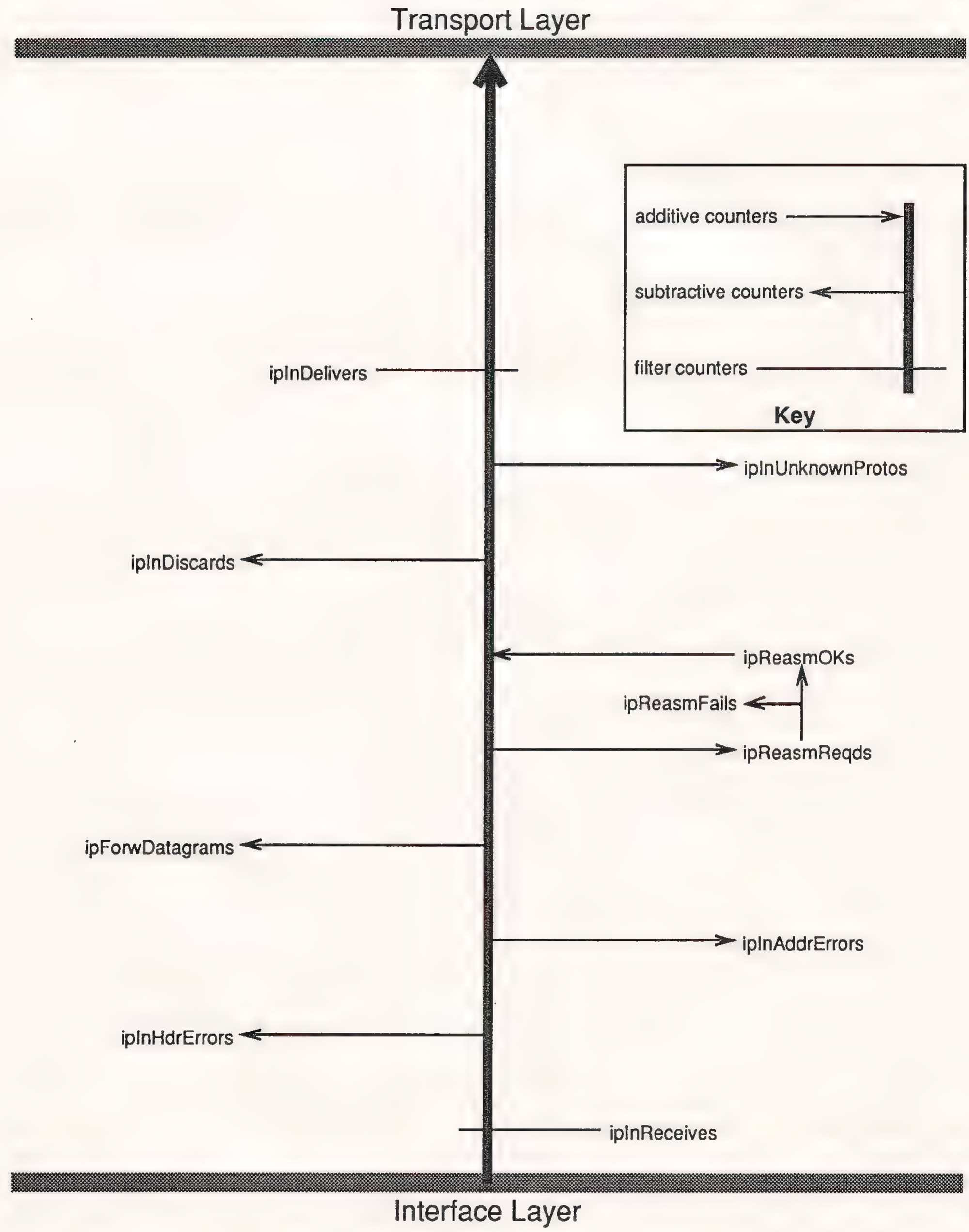


Figure 1: Case Diagram: IP Input

For example, looking at Figure 1 again, some text is required to make it clear what constitutes an IP header error versus an addressing error.

Conclusions

We believe that Case diagrams are a first step towards the problem of diagraming management information bases. We hope that their obvious benefits will encourage further research to develop better diagraming methods.

Acknowledgements

We would like to acknowledge the assistance and advice of the members of the Internet Engineering Task Force MIB Working Group.

References

[1] McCloghrie, K. & Rose, M., "Structure and Identification of Management Information for TCP/IP-based internets," RFC 1065.

[2] McCloghrie, K. & Rose, M., "Management Information Base for Network Management of TCP/IP-based internets," RFC 1066.

[Reprinted with permission from ACM SIGCOMM *Computer Communication Review*.]

CRAIG PARTRIDGE received his B.A. from Harvard University in 1983, and has been a part-time Ph.D. candidate there since 1987. For the past five years he has worked for BBN Systems and Technologies Corporation (formerly BBN Laboratories) on a variety of networking related projects including CSNET, the NSF Network Service Center (NNSC), and various projects concerned with distributed systems, IP transport protocols, and network management. In addition, he is a member of the Internet End-To-End Task Force, the Internet Engineering Task Force, and the Distributed Systems Architecture Board Task Force on Naming. He currently splits his time at BBN between CSNET, the NNSC, and managing a small TCP/IP networking project. Craig is also the editor of ACM SIGCOMM's *Computer Communication Review*.

JEFFREY D. CASE is an associate director of the University of Tennessee Computing Center where he serves as chief engineer and is responsible for engineering, local area networks, systems development for VAX/VMS and UNIX, and the University's statewide network. He is also an associate professor in the Department of Computer Science, where his principal areas of research are networking and network management in addition to his duties as manager of the Computer Science Laboratories. He is presently chair of the Network Operations Centers Tools and Applications Working Group of the IETF. Jeff received B.S. and M.S. degrees from Purdue University before completing the Ph.D. at the University of Illinois in 1983.

The CMOT network management architecture

by Amatzia Ben-Artzi, 3Com Corp.

Introduction

Over the past few years, the demand for multi-vendor networking environments has grown significantly. The TCP/IP protocol suite has become the de-facto solution for interoperability until such time as OSI systems become more widely available. As a result of the explosive growth of the TCP/IP networks, their management has received more focus and attention, yet new development of protocols for TCP/IP management was not aligned with the emerging ISO standards.

This article describes the efforts of the "NetMan" group to develop a network management system that conforms to the ISO standards, while effectively using the large deployment of existing TCP networks.

The group, which represents researchers as well as many vendors which develop and/or manufacture computer communications products, set itself the following objectives:

- To define a framework for TCP/IP network management.
- To define a standard set of mechanisms (protocols) and management information, and publish them as proposed standards (RFCs).
- To provide an architecture that allows for a straight forward migration from TCP/IP to OSI.
- To conduct the specification and design work on an aggressive time table that would lead to experimental systems before the end of 1988.

What is Network Management?

There are several different ways to look at network management. The most common is to divide the functionality into several areas.

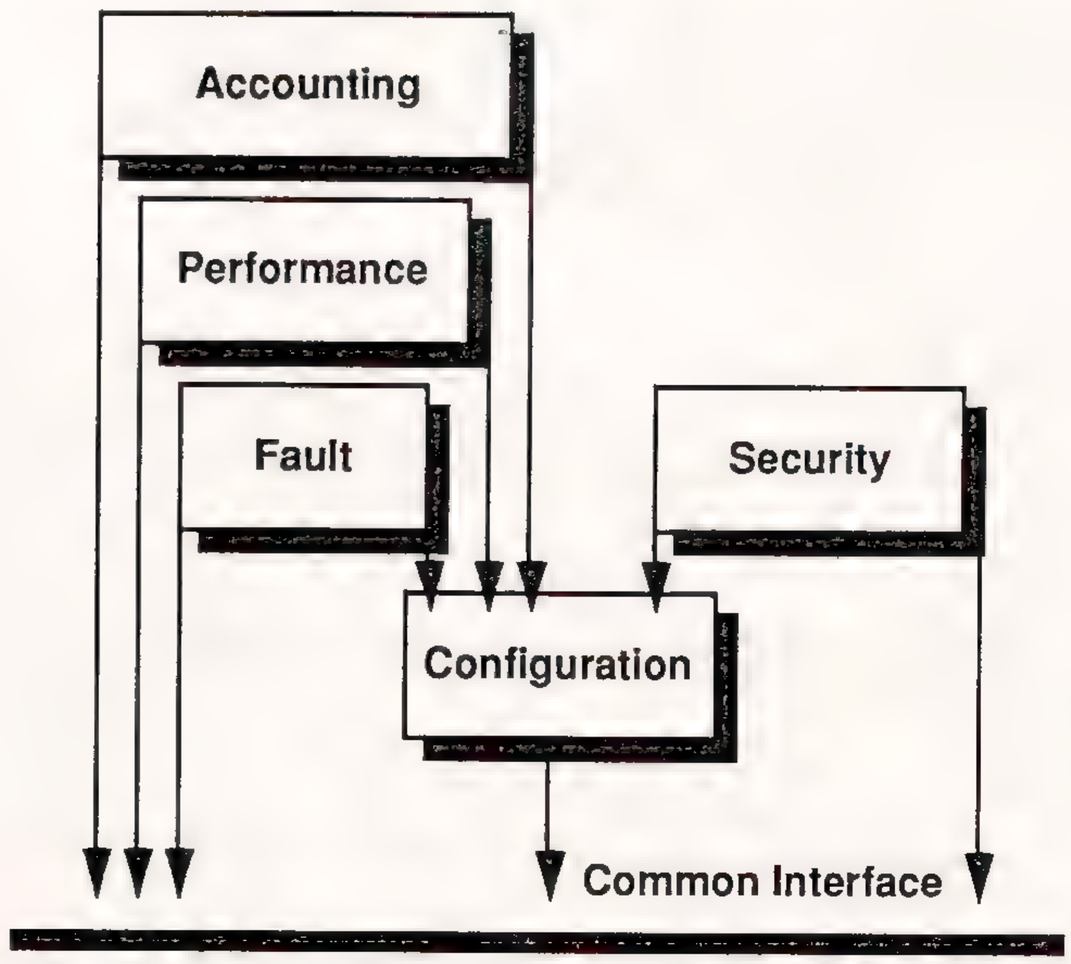


Figure 1: Network Management Applications

According to the OSI model, network management is composed of the following functional areas:

- *Configuration Management:* detection and control of the state of the network, both the logical and physical configurations of the network. Element examples are port parameters, protocol timer values, physical location of nodes, etc.

- *Performance Management:* control and assessment of the performance of a network. Element examples are throughput, error counts, retry counts, and historical statistics.

- *Fault Management:* the detection, isolation, and correction of abnormal operations in the network. Element examples are audit trail, threshold exceeded events, protocol violations, etc.

- *Accounting Management:* the collection of data related to the charges or cost of use of resources in the network.

- *Security Management:* the protection and the control of access to resources in the network. Element examples are authentication, access control, security logs, etc.

The above functional areas of management can also be treated as *network management applications*. They all depend on exchanging management data with different elements in the network, and then perform some specialized tasks based on this data. In order to maximize the benefits from the application, and reduce development complexity, it is desired that the management applications share common services, and have the same interface to the network.

CMIS

In the OSI model, a common service interface was defined for the use by all management applications. The interface provides the basic functions that a management application needs to perform its job. This interface is called *Common Management Information Services* (CMIS). CMIS provides applications with the ability to perform the following functions:

- *Monitoring:* – Get data
- *Control:* – Set data (with or without confirmation)
  – Create and delete objects
  – Perform an action (with or without confirmation)
- *Event Reporting:* – Also called "alerts" or "alarms." Information that is reported as a result of some event in the network. These events may or may not require confirmation.

Network Management models

Network management systems can be modeled in 3 different ways: By an Organizational Model, a Functional Model, and an Informational Model.

The *Organizational Model* describes how the management information is organized inside the system. It includes the naming scheme, the relations between management objects, the registration facilities, and the hierarchy used in the organization.

continued on next page

The CMOT architecture *(continued)*

The OSI model allows references to objects along two different dimensions or different "trees." The "registration tree" defines the management objects as "templates" as they may be registered by different standard bodies (such as the protocol-layer groups), while the "containment" tree defines the actual location or identification of such a "registered object" in a given network or organization.

For example, a "retransmission timer" of a transport protocol may be defined in the registration tree as part of a standard document, while the actual network component, in some network of some organization in some country may be the "containment" information needed in order to locate the right data.

The *Functional Model* describes what functions are performed by the different elements that participate in the management of networks. The most basic functional model divides the management components into *managers* and *agents* (see Figure 2). The agents are the devices that collect the information, perform commands and execute tests, while the managers receive data, generate commands, and instruct agents to perform actions, all under the supervision of an application. Such an application may be anything from a human operator (with the appropriate human interface) to a very sophisticated artificial intelligence program.

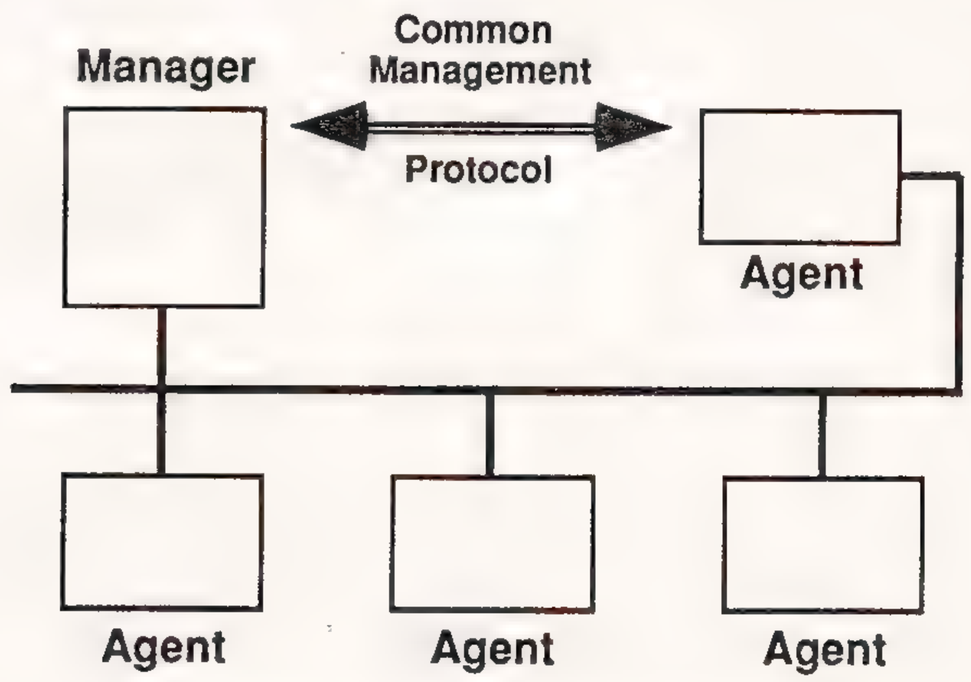


Figure 2: Functional Model

Figure 3 further describes both agents and managers as a collection of components that carry specific management information per each communication layer in the system (*Layer Management Entities*, LMEs) and a process that coordinates the activities of the different LMEs (*System Management Application Process*, SMAP).

CMIP The SMAPs are also responsible for communication between two different systems, i.e., information from an LME in a system to another system is forwarded to the respective SMAP and then communicated to the other system over the *Common Management Information Protocol* (CMIP). The functional model also allows management information to go directly from one layer in one system to the same layer in another system. However, in this case, the information is considered as management specific to a given layer (or more precisely "n-layer management") and not as part of the system management.

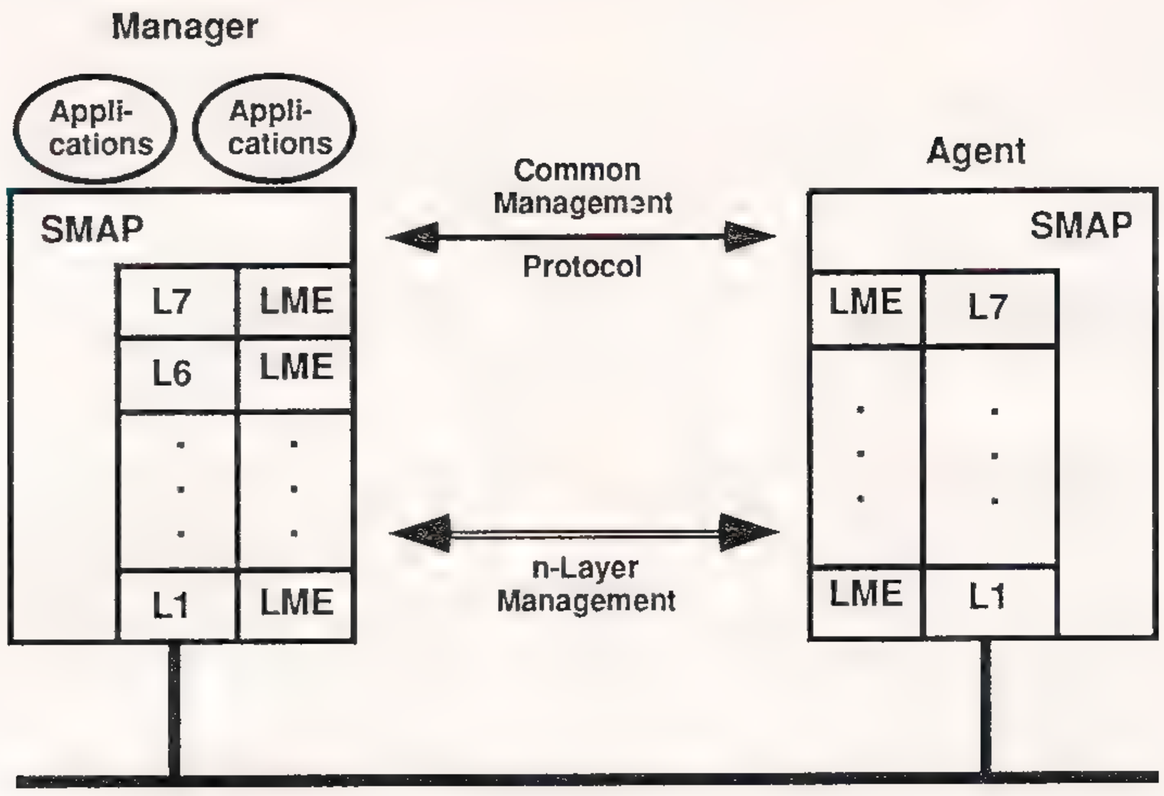


Figure 3: Detailed Functional Model

The *Information Model* refers to the way management information is represented inside the system. This is perhaps the most important element in information management in general and in interoperability in particular. The information is defined in both syntactic and semantic terms. Without such precise definitions of the management information, no "understanding" of what the information means can be established between managers and agents, even if the protocols allow for communications between the systems.

SMI

The informational model goes further and defines the different "types" of information that are needed for management information, so that the different standard bodies that define management information will have guidelines to use when they define new management information objects. These "guidelines" are called *Structure of Management Information* (SMI), and include such types as Counters, Gauges, Thresholds, Tidemarks and Rates.

From the goals of the NetMan group, it was clear that the OSI model for management had to be used, in particular, the CMIS interface. The interface for management applications was to be kept exactly as it is in the ISO standard so that investment in these applications will not be wasted in the migration from TCP to OSI.

The model for TCP/IP networks

The system developed is called *CMOT*, or "CMIP over TCP," and according to the three models for management described above, it is based on the following architecture:

Organizational Model: The management information is registered under the ISO registration tree. It uses the value assigned for the U.S. DoD as the root of the sub-tree. The tree is further broken into "management," "directory," "experimental," and "private" (Figure 4). The collection of all the information about management objects in the TCP system is called the *Management Information Base* (MIB), and is periodically updated and enhanced as new management objects are identified and defined. The current set of management information in the MIB includes 116 objects and is defined in RFC 1066. The authority for registration of information objects in the MIB is delegated by the IAB through the RFC process.

continued on next page

The CMOT architecture *(continued)*

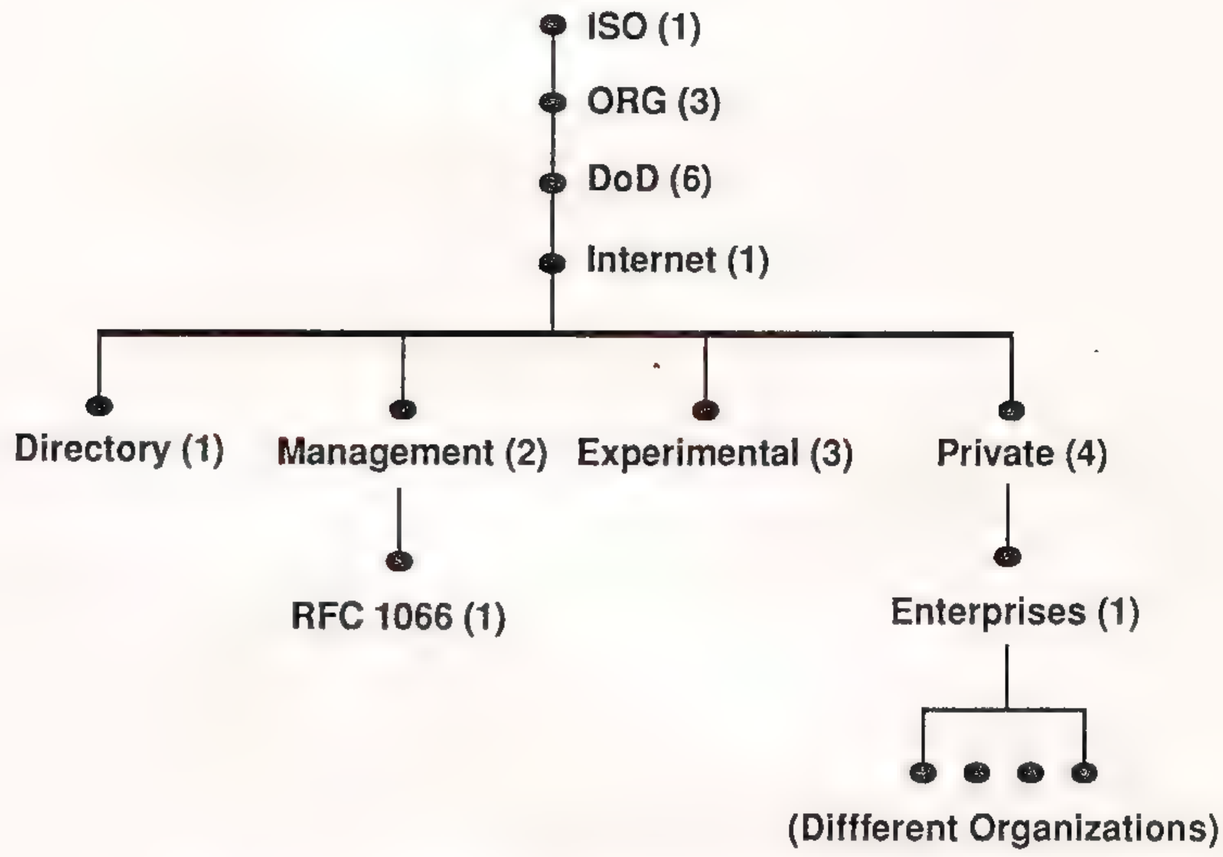


Figure 4: CMOT Organizational Model

Functional Model: CMOT adopted the OSI model. The world of management is divided into managers and agents both using their SMAPs to communicate management information. However, there is one important extension in the CMOT model over the (current) OSI model. At this point in time, ISO management can occur only over fully established connections between the managers and the agents. In contrast, the CMOT model allows management information to be exchanged over connectionless (datagram) services, i.e., the User Datagram Protocol (UDP). It is expected, however, that the ISO model will be extended in the future to allow datagram type transactions as well.

Information Model: as stated earlier, all management information is registered with the NIC so that no ambiguity or duplication of information definitions can occur. In addition, CMOT adopted the SMI model of OSI (or rather a subset of it that was stable enough to adopt), and defined the data types that can be used for defining TCP/IP management objects. This SMI is documented in RFC 1065.

One of the major conflicts that the NetMan working group faced early on during the architecture definition, was the following: On the one hand the system developed should provide management capabilities for TCP/IP networks with the goal of easy migration to OSI, but on the other hand the OSI network management standards are defined exclusively over ISO standards (such as *Association Control Service Element*, ACSE, and *Remote Operations Service Element*, ROSE).

CMOT architecture

In order to solve this conflict, the following architecture was developed: CMOT uses the ISO network management in its fullest. CMIP, ACSE, and ROSE are used per the ISO definitions. For transport, as well as the layers for which LMEs are defined, CMOT uses TCP and UDP. In between transport and application, a new layer was defined which uses a *Light Weight Presentation Protocol* (LPP) (see Figure 5). LPP provides the ISO application layer with the presentation services it expects and maps all the service calls to/from TCP (or UDP).

In addition, LPP was stripped from any presentation function that exists in the ISO presentation standard that is not used by the ISO management applications. The result was a compact layer that utilizes the TCP/IP transport, while preserving the ISO presentation services so that ISO management applications can be used with no modification. LPP is documented in RFC 1085.

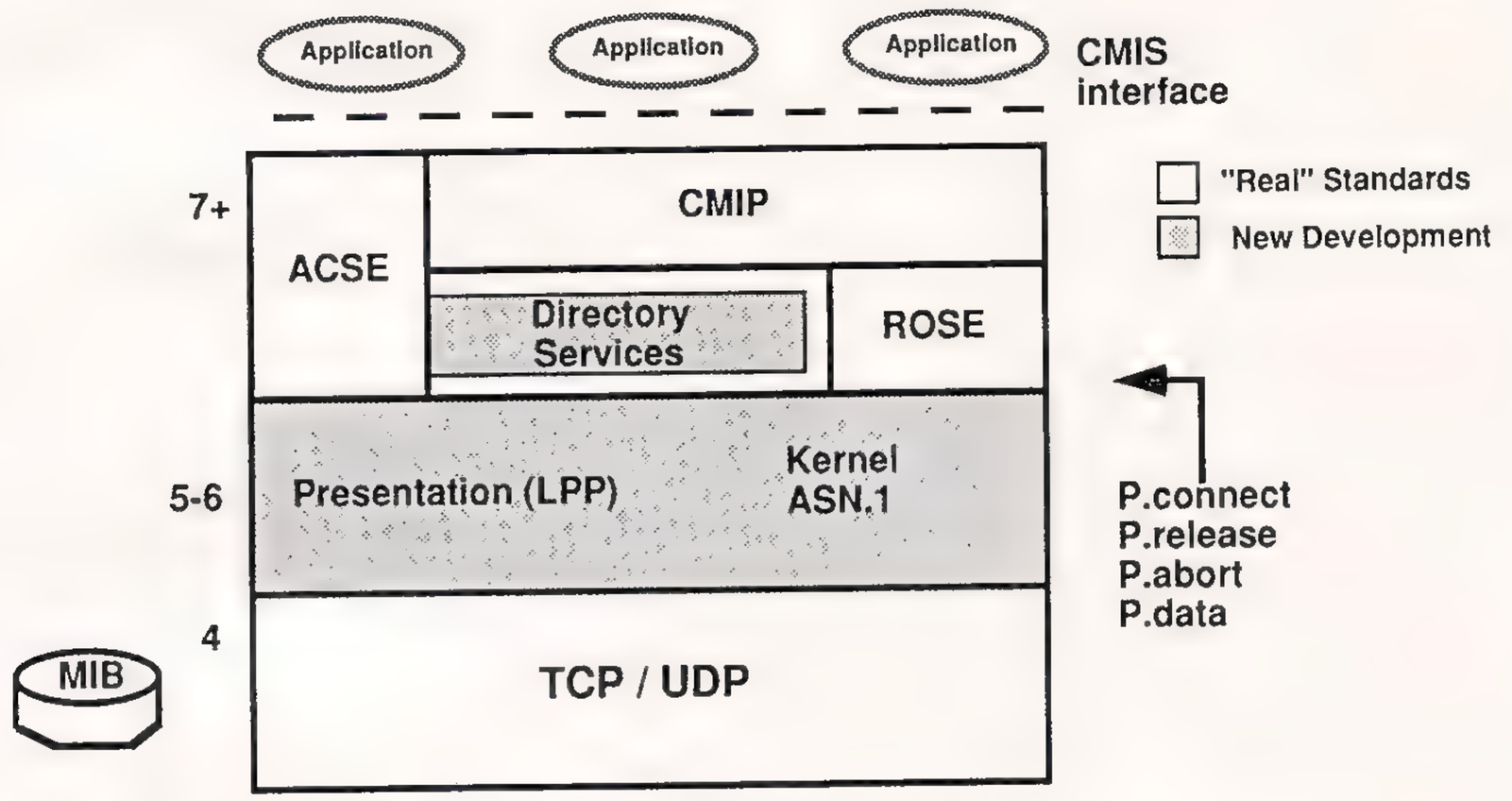


Figure 5: CMOT Architecture

Current status

In September, 1988, the first experimental system was demonstrated in a multi-vendor network environment. (See following article) This system, documented as "IDEA 0025," was developed with the major motivation of "feasibility demonstration." Some important features of the architecture (such as scoping and filtering) were left out in order to keep the first phase simple. It was also based on the Draft Proposal standard (DP) of CMIP, which has since been changed and upgraded to Draft International Standard (DIS). The demo was very successful and proved that CMIP can be implemented on a variety of platforms. Thirteen major network equipment vendors demonstrated interoperable systems, many of which were developed completely independently of each other.

During the time between September, 1988 and January, 1989, the CMOT specification was extended to reflect changes in CMIP (DP to DIS). The missing features from the feasibility demonstration were added, and additional management objects needed for the operation of CMOT were also defined.

The final touch on the standard was done during the IETF meeting in Austin, Texas, January 1989. Following the meeting, it was submitted for consideration as an RFC. CMOT, like TCP/IP, is expected to become in the near future (maybe as early as 1990) the standard in large networking environments where multi-vendor interoperability is a necessity.

AMATZIA BEN-ARTZI is Director of Systems Architecture at 3Com's Software Products Division. Previously he worked with Sytek where he was one of the major developers of their TCP/IP product line. He participated in the NetBIOS effort leading to RFC 1001 and RFC 1002, and is active in the CCITT/ISO standardization work. Amatzia authored the CMOT architecture and is active in the NetMan group. He received a B.Sc. in Mathematics from the University of Tel-Aviv, and an M.Sc. in Computer Science from San Jose State University.

The NetMan demonstration at INTEROP™ 88

by George Marshall, 3Com Corporation

INTEROP™ 88 was held September 26–30th, 1988 in Santa Clara, California. This was the third annual INTEROP, and the first one with a exhibition hall in addition to the technical tutorials and conference sessions. [See also "INTEROP 88 Conference Report," in *ConneXions* Volume 2, No. 11, Nov. 1988, and "The INTEROP 88 Network—Behind the scenes" in *ConneXions* Volume 3, No. 2, Feb. 1989 —Ed.]

One of the highlights of the show was the "NetMan" demonstration: thirteen companies presented a joint demonstration of network management with true interoperability.

The NetMan Group

"NetMan" is the name of a network industry group which is defining long-term network management protocol standards for TCP/IP networks, according to the directions given by the IAB in RFC 1052. NetMan is a working group of the IAB Internet Engineering Task Force (IETF).

Using the system defined by the NetMan group, network administrators can manage network equipment from many vendors from the same console. The approach used is to adapt the CMIP (Common Management Information Protocol) layer for OSI to operate above the standard TCP/IP stack; the resultant combination is referred to as *CMOT* (CMIP over TCP). The use of OSI application protocols provides an easy transition to all-OSI networks.

Participants

The NetMan project includes active involvement by major TCP/IP networking companies: 3Com, CMC, Convergent Technologies, Digital Equipment Corporation, Excelan, Hewlett-Packard, MITRE, Network General, Sun Microsystems, Sytek, Ungermann-Bass, and Unisys.

Equipment

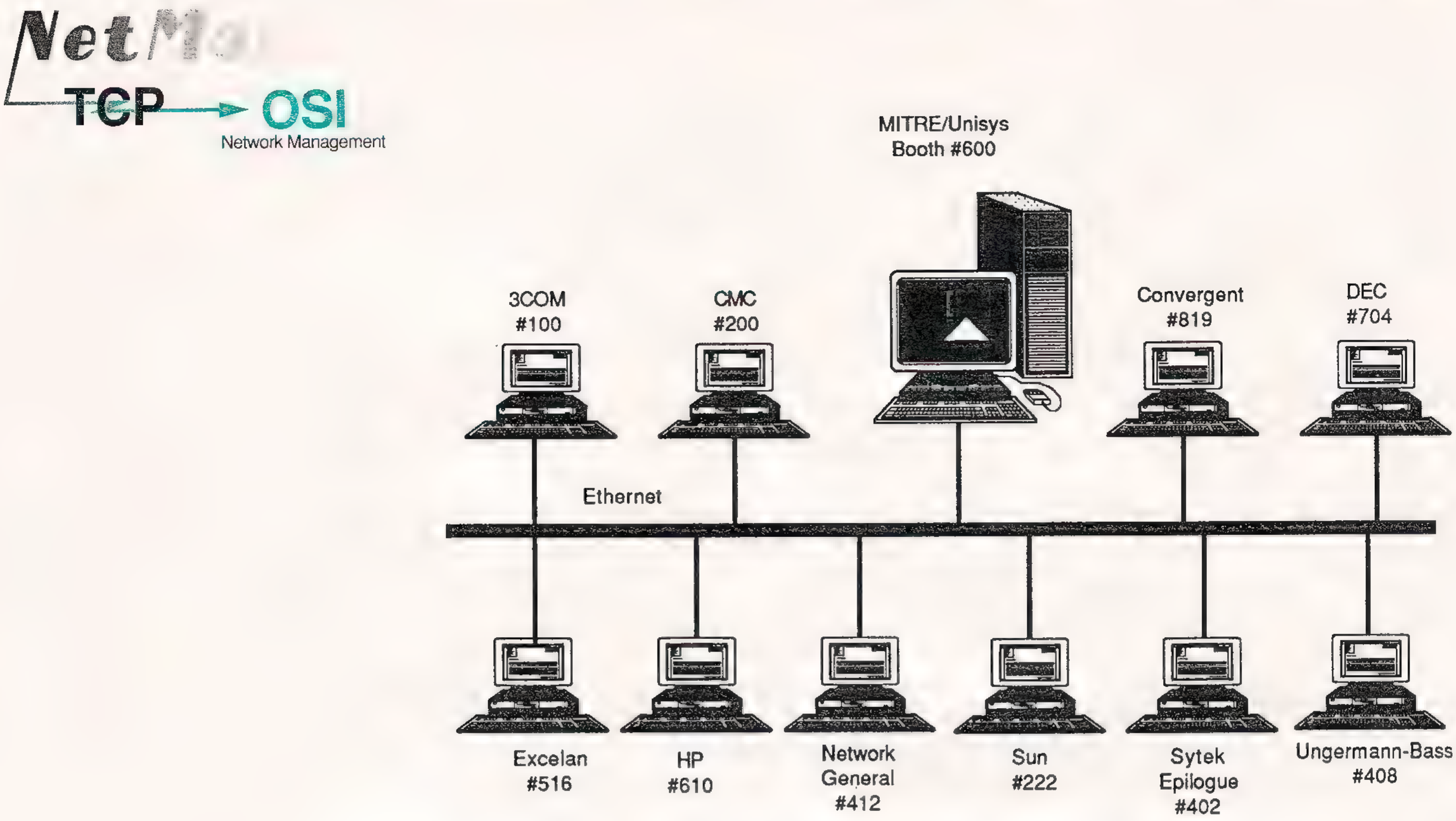
The demonstration itself included a central net management console, jointly sponsored by all participants, with network equipment from each vendor participant being managed by the central console. The central manager ran continuously during the show, managing the "agent" nodes, retrieving and plotting information from each agent system. The systems being managed in the individual booths included terminal servers, PCs, workstations, and minicomputers.

In addition to the central demonstration, 3Com and HP also demonstrated manager stations in their own booths, managing other agents on the show floor.

A first step

The demonstration at INTEROP 88 was a feasibility demonstration, intended to show the viability of the CMIP over TCP approach to standard network management products; most vendors showed prototypes or concept demonstrations, rather than products. Most vendors participating in the demonstration also indicated their intent to introduce products using CMOT.

GEORGE MARSHALL is Product Marketing Manager for Network Management at 3Com's Enterprise Systems Division.



INTEROP 88 NetMan demonstration.
(Logo courtesy of Sytek, Inc., diagram courtesy of Communication Machinery Corporation.)

Symposium on Network Management

The First *International Symposium on Integrated Network Management* sponsored by IFIP WG 6.6 and hosted jointly by the National Institute for Standards and Technology (NIST) and MITRE Corporation will be held in Boston, May 14–17, 1989. The objective of the symposium is to create an international forum for information exchange and cooperation between vendors, system integrators, users, researchers, and standardization bodies. In particular, the program of the symposium will concentrate on the following subjects, emphasizing the integration of different systems:

- Management requirements and standardization issues
- Fault, configuration and name, accounting, performance, and security management
- Quality of service
- Knowledge-based systems
- User interfaces and management languages
- Models/architectures/algorithms
- Heterogeneous networks
- Protocols
- Management data bases
- Planning systems
- Implementations and case studies

For more information contact Hershey Young, NIST, 301- 975-3600.

Network Management and the Design of SNMP

by Jeffrey D. Case, University of Tennessee at Knoxville
James R. Davin, MIT Laboratory for Computer Science
Mark S. Fedor, NYSERNet Inc.
Martin L. Schoffstall, NYSERNet Inc.

Introduction

With the emergence of wide area networking, a decentralization of network infrastructure, and the increasing interconnection of previously isolated networks, there came a pressing need for network management solutions that transcend the scope of single local area networks. The *Simple Network Management Protocol* (SNMP) was based on a unique collaborative effort between users of networks, university researchers, and communications vendors to answer this need. After almost two years of operational experience with multiple independent implementations, SNMP was adopted as the Internet standard for management of the full range of network devices—routers, hosts, terminal servers, bridges, PCs, and other communications gear.

Current status

The credibility of SNMP derives in part from the methodology that shaped it—a methodology that recalls the emergence of the original TCP/IP protocols. Rather than conceding to an increasing trend of formal specification without the benefit of real-world verification, SNMP was engineered according to a cooperative discipline that emphasized practical, empirical research and iterative testing. The prototypical ancestor of SNMP (described in RFC 1028) was implemented by three independent groups, tested for interoperability, and refined by wide operational experience. The results of this extended process are represented in three documents that capture the substance of TCP/IP network management: RFC 1065 describes the structure of management information; RFC 1066 describes the actual content of the management information made available at each node; RFC 1067 describes the SNMP itself. The technical confidence of the Internet community in the SNMP approach is reflected in the policy statements of RFCs 1052 and 1083.

SNMP is widely deployed in the Internet today to manage complex internetworks at large corporate and university sites, and within regional networking consortia. The popularity of SNMP as the network administrator's choice lies in its proven heterogeneous interoperability and its wide availability. Over 20 independent commercial implementations are available off the shelf, with at least two public domain implementations available from MIT and CMU. Consistent with the traditional openness of the TCP/IP architecture, SNMP is available in the form of complete, turnkey management solutions as well as source code libraries from which unique applications can be easily constructed.

The SNMP Design Philosophy

Much of the technical foundation of SNMP is articulated in a single observation: network management is an application fundamentally different in its requirements than any other application that makes use of the network. This article describes how the important differences between network management and other applications have shaped the design of SNMP, in particular, how special requirements of ubiquity and robustness are satisfied by the SNMP paradigm.

Design for omnipresence

The requirement of ubiquity is one difference between network management and more common network applications. While the essential value of services like Telnet or FTP is hardly diminished by their absence in certain quarters, effective network management seems to require nearly universal deployment of the management mechanism. In this respect, the role of a network management protocol is comparable to that of ICMP (Internet Control Message Protocol) in the Internet architecture: it represents the basic mechanism that all individual nodes must have to ensure a smoothly functioning whole.

The design of SNMP is very much "optimized for omnipresence," for, although the ubiquity question can be answered in a purely legislative way by standards agreements, the peculiar engineering realities of network management can impose inordinate costs on such a course.

The instrumentation problem

One of the less obvious realities is the cost of network management instrumentation—compounded by both the "invasive" character of the management function and the heterogeneity of today's large networks. In order to be effective, the network management agent at each node must have access to counters and error indications down to the lowest layers of the network. This need for intimate knowledge of hardware and software internals means that, in general, the engineering costs of instrumentation cannot be shared among different types of network devices. And, yet, the ability to deploy a variety of technologies to address different communications needs is the cornerstone on which the most cost-effective networks are built.

This unyielding dilemma figures largely in the design of SNMP. For the sake of cost-effective, ubiquitous deployment, SNMP imposes the fewest possible constraints on the instrumentation costs for each new type of network device. The design of SNMP centers upon a few scalar data types, familiar to applications developers, and easily represented by many machines. Moreover, the SNMP design minimizes the cost of instrumentation by divorcing the internal organization of information in a network device from its representation in the protocol. Complex, aggregate data structures that require an engineer either to assemble them anew for each management request or to shape critical data structures in potentially suboptimal ways are deliberately omitted from SNMP. Instead, the simple ordering relationship by which SNMP organizes management information admits implementation in a technology-independent way, and contributes little to the repeated engineering costs of instrumentation.

The cost of management systems

Less obscure, perhaps, than the problem of instrumentation is the question of overall development costs for a truly ubiquitous management solution. Controlling the costs of network management may be a little like containing the Federal deficit: progress is more likely to come from tough engineering decisions than from legislated compromises. The design of SNMP attacks this problem at several levels simultaneously by assigning the largest share of management responsibility to the management stations—rather than to the managed nodes.

continued on next page

Design of SNMP *(continued)*

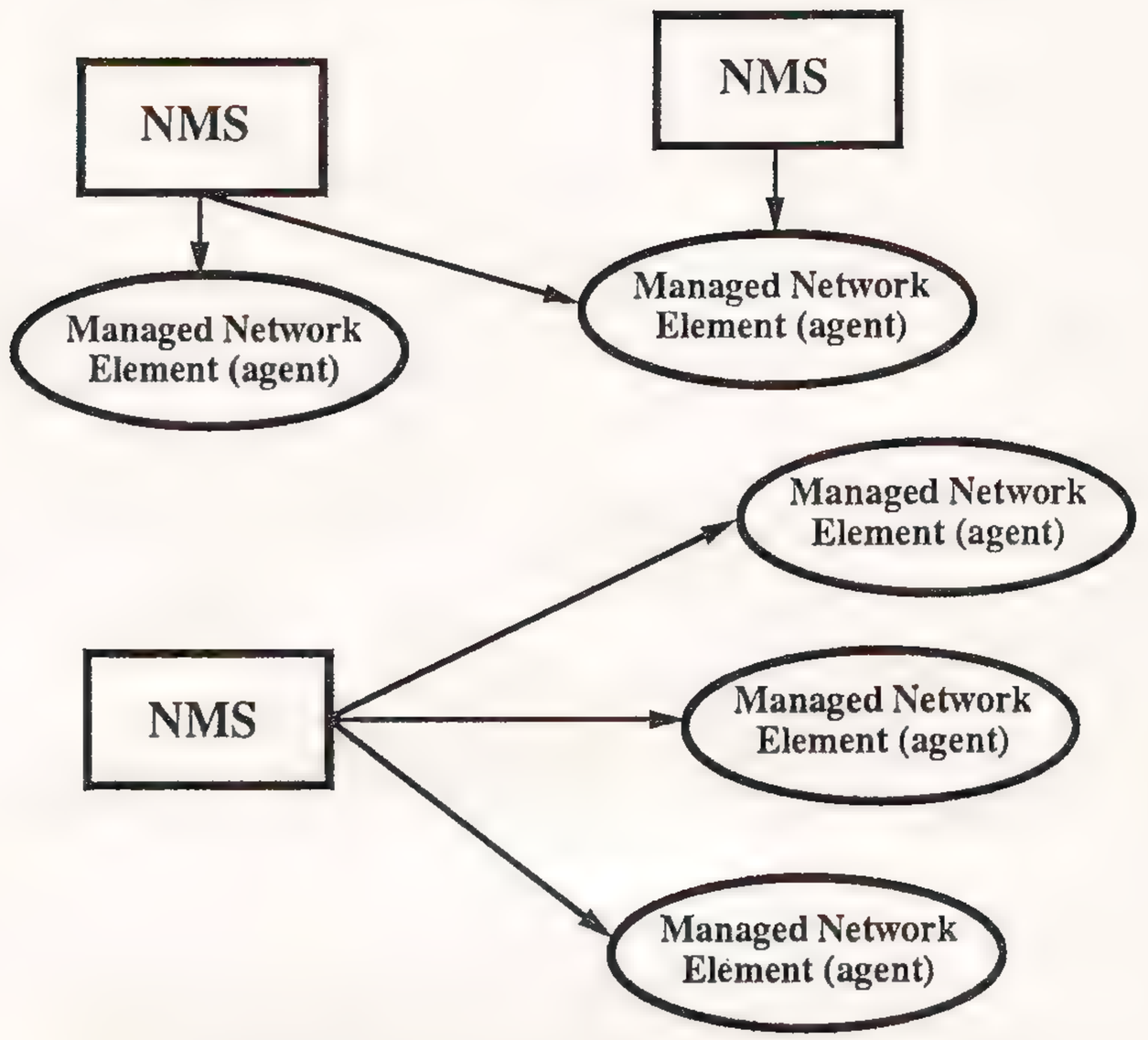


Figure 1: SNMP System Model

On one level, SNMP's rigorously minimalist design reduces memory and CPU resources required for its operation—with the result that the per-node cost of network management is lower with SNMP than with more elaborate mechanisms. SNMP's data access techniques are of logarithmic—not linear—complexity. And its well-engineered subset of ASN.1 minimizes the CPU burden commonly associated with more lavish use of these encodings.

On another level, the economy of the SNMP design admits its deployment in inexpensive devices where more elaborate mechanisms just don't fit. From tiny Kinetics boxes and PCs, to gargantuan Cray processors, SNMP today enjoys the truly ubiquitous deployment required for effective network management.

On a third level, the SNMP strategy of shifting the management burden away from the managed nodes not only accrues per-node economies from simplified implementations, but it also exploits the numerical predominance of managed nodes in today's networks: by limiting the cost of the more common component, SNMP affords correspondingly greater reductions in overall systems cost.

The success of the SNMP approach is not so hard to understand when considered as part of a more global technological trend: much as the combination of simplified RISC processors with increasingly sophisticated compiler technology has produced powerful computing engines, so the combination of simplified protocol mechanisms with increasingly intelligent management stations has produced a powerful tool for network management.

Design for robustness

When all else in the network fails, network management, if at all possible, must continue to function. The robustness requirements of network management distinguish it from a wide range of less exacting applications and poses a difficult—if not impossible—challenge to the protocol designer.

On one hand, no protocol is likely to transcend a complete network collapse; on the other hand, it is also true that the robustness of a management protocol in times of network stress is lamentably unrelated to the authority of standards bodies and remarkably difficult to test in isolated network environments. Accordingly, conservative engineering and attention to operational experience are among the best strategies for realizing the extraordinary robustness required for effective management.

The conservative engineering discipline implicit in SNMP is one that, in the automotive world, might produce a four-wheel drive pickup instead of a limousine. Although the limousine is undeniably more comfortable and stylish for cross-town jaunts to social or business engagements, the pickup not only serves adequately on those occasions, but it is also designed to cope with payloads and road conditions to which the more pricey limo is not suited.

In much the same way, while SNMP serves adequately for routine accounting and configuration tasks in times of network health, much of its design is shaped by the demands of "network fire-fighting"—diagnosing and repairing problems in an operating internet.

Holding the road

Part of the "all-terrain" character of SNMP lies in its conservative demands upon underlying transport mechanisms. SNMP is designed so that it can operate over UDP, TCP—or even link layer or MAC layer protocols. This broad adaptability derives largely from SNMP's simplified request-response paradigm. When used with connectionless transports, this paradigm reduces the overall exposure of each management operation to failures arising from corruption and loss of individual packets. In the presence of error-prone links, SNMP's superiority to Telnet-based management techniques is explained largely by this aspect of its design. Moreover, when a management operation does fail as the result of packet loss, the decision to retry the operation—or, based upon the failure, to shift to a more promising global strategy—is assigned by SNMP to the management application itself rather than to the underlying transport service.

SNMP's focus on tight control of network resources is also manifest in its restriction of unsolicited management traffic from managed nodes. The SNMP strategy of "trap-directed polling" identifies only the most critical network conditions—outright link failures or early signs of a viral attack—as meriting unsolicited management traffic. In addition, management operations that could unexpectedly elicit long streams of packets from a managed node are precluded by SNMP. While operations of this sort may at first appear like an attractive convenience, their cost in terms of lost control and additional protocol mechanism is unjustified: reliable delivery of such a packet stream entails the cost of a reliable transport; unreliable delivery guarantees the manager no more than partial, potentially misleading information.

Carrying the load

If robustness requires operation in a wide range of network "terrains," then it also requires accommodation for an unpredictable range of management information "payloads."

continued on next page

Design of SNMP *(continued)*

The organization of management information within SNMP means that neither individual management applications nor overall applications design is restricted by the preconceived information groupings defined in standards agreements. Rather, each SNMP protocol operation can address an arbitrary set of management variables—crafted to suit immediate operational demands. Operational robustness is also enhanced by the powerful SNMP get-next operation. Aside from its routine use to organize management information in a tabular way, it affords a simple mechanism by which applications designers can cope with version skew, omissions, or other irregularities in the management information base.

Conclusion

In times of network health, SNMP is probably quite similar to other management strategies in its effectiveness and overall resource use; in times of network crisis, however, the conservative design of SNMP suggests it as the network management strategy of choice. Its operational robustness, together with its economical ubiquity, are perhaps why the roving eyes of network managers have focused on SNMP.

JEFFREY D. CASE is an associate director of the University of Tennessee Computing Center where he serves as chief engineer and is responsible for engineering, local area networks, systems development for VAX/VMS and UNIX, and the University's statewide network. He is also an associate professor in the Department of Computer Science, where his principal areas of research are networking and network management in addition to his duties as manager of the Computer Science Laboratories. He is presently chair of the Network Operations Centers Tools and Applications Working Group of the IETF. Jeff received B.S. and M.S. degrees from Purdue University before completing the Ph.D. at the University of Illinois in 1983.

JAMES R. DAVIN is currently on the research staff of the Advanced Network Architecture group at the MIT Laboratory for Computer Science. He was previously a member of the router development group at Proteon, Incorporated, where much of his work focused on network management. Before joining Proteon, he worked on a variety of communications protocols for the Advanced Systems Development group at Data General's Research Triangle Park facility. He holds the B.A. from Haverford College and masters degrees in Computer Science and English from Duke University.

MARK S. FEDOR is a Project Leader in the Research and Development group of NYSERNet Inc. in Albany, NY. For the past year and a half, he has been involved in the technical aspects of the New York State Educational and Research Network as well as the development of network management software. Prior to joining NYSERNet, he worked at the Cornell Theory Center where he took part in the development and operation of the NSFNET backbone. He was also the primary developer of the *gated* routing daemon for UNIX. Mark is a member of the Internet Engineering Task Force where he is involved in the Open IGP working group which is charged with developing a new routing protocol. He received a B.A. in Computer Science from State University of New York, College at Oswego in 1986.

MARTIN L. SCHOFFSTALL, known to his friends as "The Lumberjack of Network Management," is a researcher and adjunct professor at the Rensslear Polytechnic Institute. He is also the Director of Technology for NYSERNet Inc. Marty was previously a Systems Engineer for BBN in Cambridge, MA. He is a member of the Internet Engineering Task Force where is is chairman of the Authentication Protocol working group. He holds a B.S in Electrical Computer and Systems Engineering from RPI.

The UCL Project INCA network management system

by **Graham Knight, Department of Computer Science, University College London**

Introduction

University College London has taken part in the INCA project [1] which was partially funded by the European Commission under the ESPRIT programme. (The INCA Participants were: Modcomp Computer GMbH, GEC Research Ltd, Nixdorf Computer, Olivetti, and University College London). INCA has carried out studies, and begun implementation, of a set of integrated office services operating above a set of interconnected networks. The project adhered firmly to OSI principles in its work and consequently the management of OSI networks was of concern. A close interest was taken in the developing ISO network management standards. These standards were seen as enabling a platform to be built, on which generic management tools could be constructed.

Of prime concern to us was the information model that was being developed by ISO [2][3]. This adopts an object oriented approach and will eventually specify the attributes of a set of *Managed Object Classes* (MOCs) and the relationships between *Managed Objects* (MOs). ISO calls the complete collection of management information conforming to this model a *Management Information Base* (MIB). The objective of the piece of work described here was to define a schema for a limited MIB and to implement management agents and clients above the *ISO Common Management Information Service* (CMIS), in order to test out our ideas. It is assumed that the reader is broadly familiar with the shape of these emerging ISO network Management Standards.

The Management Information Base

A number of questions arose in discussion during the MIB design phase: what was an appropriate set of MOCs, what should be their attributes, how should MOs be named? How should transitory objects such as connections be handled? What were the correspondencies between the OSI abstractions which were modelled by the MOs and real objects in real systems?

It was felt that practical experience would be valuable in answering these questions. In order to keep the extent of the practical work within bounds we have concentrated exclusively on the management of the transport layer (specifically the ISODE [4] transport layer), with emphasis on the needs of performance and fault management. There are two main reasons for this choice: firstly, the ISODE transport layer is in user space and the sources of the code were available to us; secondly, the transport layer is especially important in the context of an integrated network as it is the lowest layer to provide an end-to-end service.

Managed Object classes

It seemed obvious that generality could only be gained by ensuring that the MOCs we defined corresponded to abstractions from the OSI model. This seemed to be the tacit assumption within the standards groups, although no very firm statements were available. At the same time we wanted there to be a strong correspondence between the abstract MOs and the real components that were familiar to users of the ISODE software.

continued on next page

The INCA network management system *(continued)*

Our final choice of MOCs was as follows:

- *Managed System:*
  This corresponds in our case to "a UNIX system."

- *(N)-Subsystem:*
  It seemed natural that the first level of decomposition presented to the user below the *Managed System* should be seven *(N)-Subsystem* MOs, hence the need for a *(N)-Subsystem* MOC.

- *(N)-Entity:*
  A set *(N)-Entities* is the result of a relatively static partitioning of the *(N)-Subsystem*. For us, the most useful partition would be along lines determined by boundaries between implementations. Thus, had we two separate *T-Layer* implementations, we would have two *T-Entity* MOs. As it happens, we have only one, so the niceties of this distinction are somewhat lost!

- *(N)-Entity Invocation:*
  The relationship between Entity, Entity Type, Entity Invocation, and Connection is not very clear from the standards, and has shifted with time. However, in the UNIX environment, it is attractive to interpret the *(N)-Entity-Invocation* as a UNIX process.

- *(N)-CEP*:
  It is here that most of the counters and gauges which measure activity are located. Counters in the other objects are often aggregates of these.

In addition to the MOCs given in the table above, there are several MOCs which exist solely for management purposes. Examples of these are the threshold and event report control MOCs.

Naming

The order in which the MOCs were introduced above is no accident. It corresponds to the hierarchical relationship between objects which is used for naming. Naming follows the X.500 Directory "distinguished name" scheme quite closely. In each object there is a `ManagedObjectIdComponent` as follows:

```
ManagedObjectIdComponent ::= SEQUENCE
  {
  componentType   <N>ManagedObjectClass,
  componentValue  <N>ComponentValue
  }
```

The name of an object instance corresponds to a path through the managed object tree. Thus

```
ManagedObjectId ::= SEQUENCE OF ManagedObjectIdComponent
```

Figure 1. shows an example hierarchy within a real system. The first two `componentValues` (*`ManagedSystemId`* etc.) are NULL, since there can only be one instance of this MOC in this position in the hierarchy. The *T-Entity-Invocations* are distinguished by UNIX process ids, and the *T-CEPs* by UNIX file-descriptors.

The full name of the underlined *T-CEP* may thus be expressed as:

```
{Managed System=NULL,T-Subsystem=NULL,T-Entity="isode",
 T-Entity-Invoc=1352,T-CEP=5}.
```

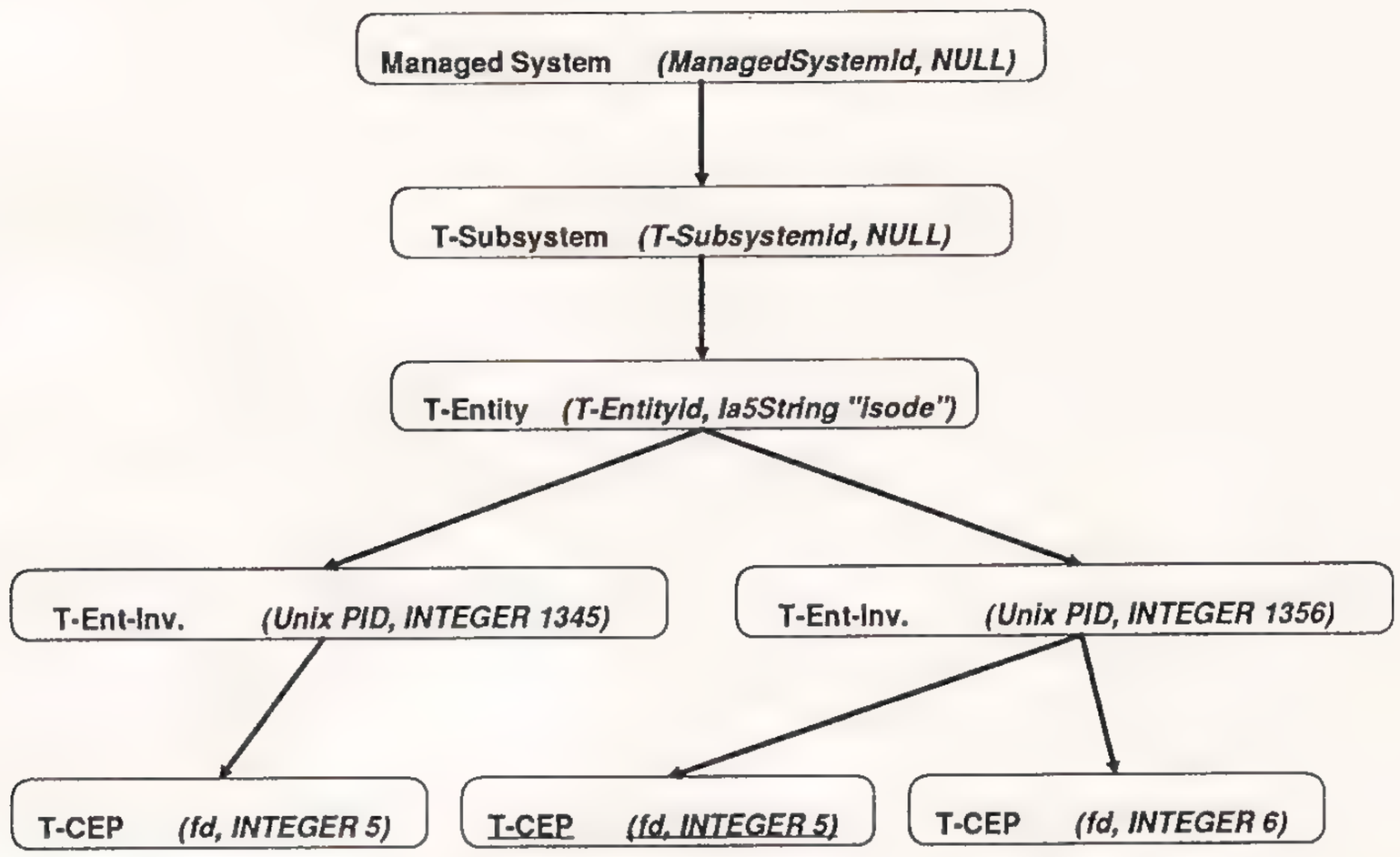


Figure 1: Example Managed Object Tree

Locating objects

The analogy with the Directory is not perfect. A particular difficulty arises in gaining access to transitory objects such as *T-CEPs*. These have identifiers which are allocated locally, and are not visible outside an open system until the management system makes them so. For example, a management process which has access to a *T-Entity-Invocation* needs to be able to obtain the identifiers of the *T-CEPs* it contains. This is analogous to the Directory "list" operation. CMIS does not specify an equivalent operation, and presumably one is supposed to rely on the "filtering" mechanism. Unfortunately, filtering was not to be included in the first implementation, so a different mechanism was needed. This has been achieved by ensuring that the `ManagedObjectIdComponents` of subordinate managed object instances are always present *explicitly* amongst the attributes of the parent managed object instance. These can then be read by a M-GET operation and hence the full names of the subordinates may be constructed.

Event Reports

Significant events, such as the triggering of thresholds, cause reports to be generated. The destination of the report for each event is governed by a Report Destination List in a "report control" MO. According to [2], destinations may be specified either as PSAP Addresses or Application Entity Titles. No guidance is given as to whether event reports should use existing management associations or create new ones.

In the UCL implementation, where destinations are specified in these two ways, an association is established the first time a report is sent, and is then used for all subsequent reports to the same destination. This allows spontaneous reporting to a "well-known" Event Logger. However, there is a problem when the association is initiated by a remote process which then wants to receive event reports across the same association.

continued on next page

The INCA network management system *(continued)*

Inserting its AET or PSAP Address in the list of destinations does not work, as it will cause a new association to be established. The UCL implementation includes a third method of destination specification. This allows a special symbol to be supplied as the destination, which implies "use the current association."

Management Agents and Clients

The previous sections have dealt with the information model employed in the UCL MIB. This section describes some of the implementations which have been built in accordance with that model. The aim was to start to build up a set of management tools which could be run from anywhere within the network. The major components are:

- *System Management Agents* (SMAs): There is one of these on each participating system. Each provides the external ISO MIS interface on behalf of their host. The design of the SMA is discussed below.

- *System Status Display* (SSD): This maintains a dynamic display of the current communications activity on a given system or set of systems. It is driven by event reports from the SMAs.

- *Event Collection Process* (ECP): This receives reports from the SMAs, filters these, and records them for subsequent analysis.

- *Microscope*: This gives a user interface to the MIB on a system, allowing all attributes of the MIB to be accessed individually. It may be used for trouble-shooting and the direct manipulation of management values such as thresholds. The Microscope is based on the SUNview software. It allows the user to request a list of subordinate MOs, select one of these with a mouse and so descend the tree. Other buttons are available to display and modify attributes.

The implementation environment is UNIX 4.2BSD running on a variety of hardware with the OSI infrastructure being supplied by ISODE. Testing has mainly been done across the Ethernet (with TCP as a "Network" layer), and over X.25. All implementation is in **C**. The ISODE presentation layer toolkit has been used to assist with ASN.1 processing.

System Management Agents

Eventually the MIB will cover all the OSI layers on a UNIX system (and maybe some non-OSI ones!). It was decided that this could best be implemented by mounting a single specialised management process on each system. This would enable OSI MIS implementation to be kept in one place and would simplify access control and synchronisation. This process is called the "Systems Management Agent."

The ISODE Transport layer, exists as protocol code bound to application processes and running in user space. Some IPC is required between these processes and the SMA, (see Figure 2). Currently UDP sockets are used, all IPC being initiated by the application processes.

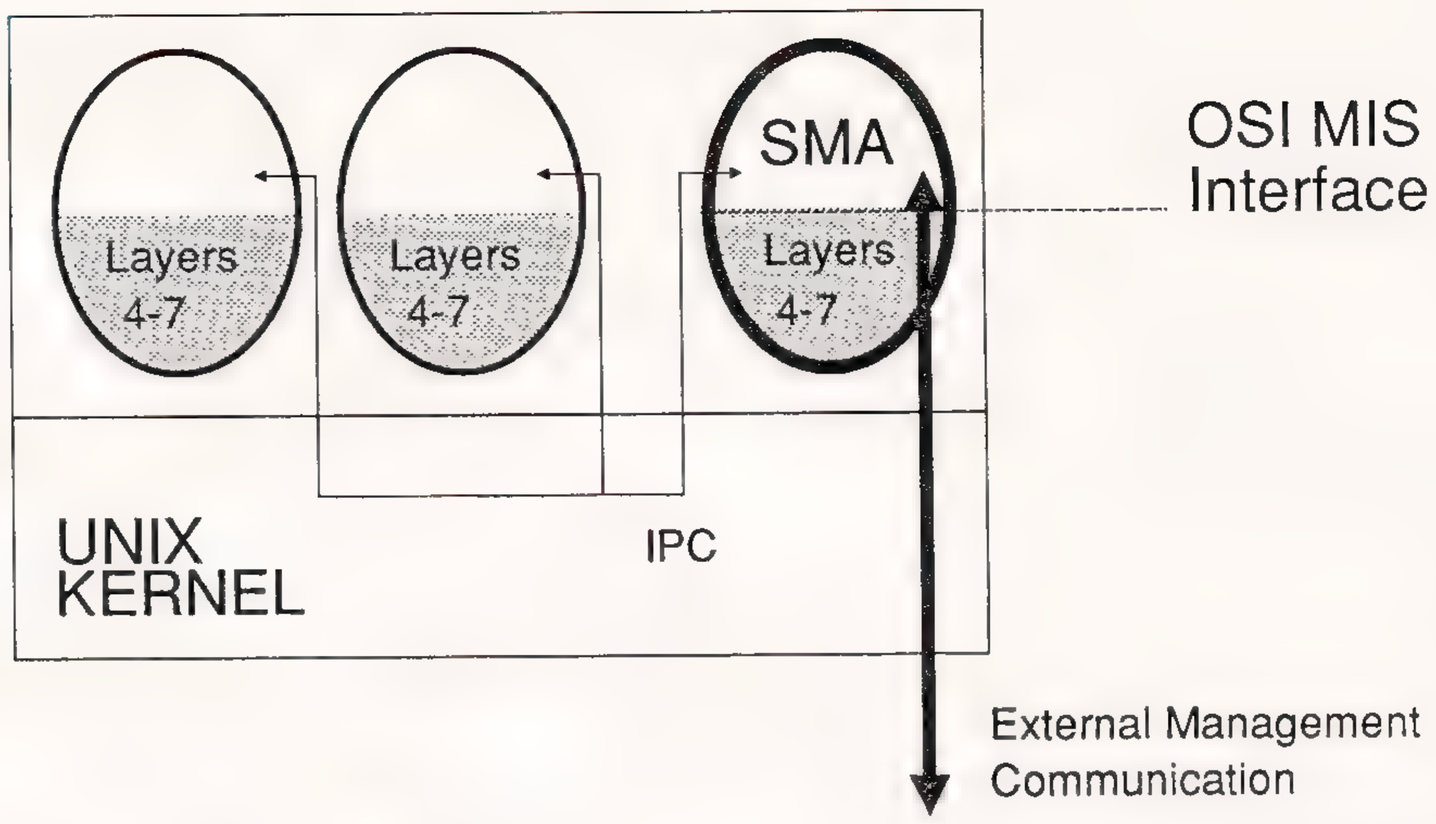


Figure 2: Management Communication within a UNIX system

The ISODE transport implementation has been modified so that it sends messages to the SMA following "significant" events. Currently, these are Connects and Disconnects, the triggering of traffic thresholds, and errors. At present, the traffic threshold is triggered by the transmission or reception of 10 T-PDUs—a compromise between timeliness and management overhead. This scheme keeps the impact on the transport layer code to a minimum and avoids problems with having the transport layer deal with asynchronous events generated by the management system. However it inevitably means that the information held by the SMA (and hence accessible externally) is always somewhat out-of-date. A second consequence is that it is difficult to arrange for information to flow from the SMA to the application process. Currently, flow in that direction is not allowed; all writable attributes being held exclusively within the SMA's own data space. Thus, whilst it is possible to control the operation of the management system itself—by setting thresholds etc., it is not possible to control the operation of the transport layer. The datagram-style IPC makes design of the SMA simpler as it is essentially stateless. However it does necessitate the use of somewhat ugly back-door methods to detect the abnormal demise of an application process.

The SMA represents managed objects internally as a series of C structures linked by pointers to reflect the hierarchical relationships. Experience from the UCL *QUIPU* Directory service implementation [5] has been useful here. All data is kept in main memory. The SMA communicates with the outside world across the ISO CMIS interface. This is provided by the CMIP protocol which is implemented on top of the ISODE ROS interface and thence on a full OSI stack.

Summary and future work

A simple ISO-style network management system has been built and is now being evaluated by users and managers. The design exercise threw up several problems associated with the information model, particularly those associated with the naming and location of information. Interim solutions have been developed. Implementation has given us some insights into the factors affecting efficient performance and the impact the management system has on communications performance.

continued on next page

The INCA network management system *(continued)*

In the future, we intend to develop the system to cover more layers within a UNIX system, and to implement mechanisms to control protocol operation. Projects are currently underway to provide proxy agents for SNMP-based systems and for nodes on an FDDI ring.

References

[1] G. Knight, G., & Kirstein, P. T., "Project INCA—An OSI Approach to Office Communications," Proceedings of the OSI87 Conference, pp 245-254, Online, London 1987.

[2] ISO TC97/SC21 WG4 N 2684, Information Processing Systems, Open Systems Interconnection—Management Information Services—Structure of Management Information, April 1988.

[3] ISO TC97/SC21 WG4 N 2685, Information Processing Systems, Open Systems Interconnection—Management Information Services—Generic Definition of Management Information (GDMI) skeleton draft, April 1988.

[4] Rose, M.T., "The ISO Development Environment User's Manual," The Wollongong Group, Palo Alto CA, July 1988.

[5] Kille S.E., "Distributed Operations in the QUIPU Directory Service," "ESPRIT88—Putting the Technology to Use"—pp 1063-1074, Elsevier, Amsterdam, Nov 1988.

GRAHAM KNIGHT gained a B.Sc. degree in maths in 1970 and taught in schools and colleges for several years before taking an M.Sc. in Computer Science at UCL in 1980. Since then he has worked in the Computer Science Department at UCL, first as a research assistant and now as a lecturer. He has been involved in several collaborative research projects in the fields of integrated networks, network management, and ISDN. Currently he is leading the UCL team in the ESPRIT project "PROOF" which is studying at the use of Primary Rate ISDN in the office.

Upcoming Events

Advanced Computing Environments will be hosting an *Internet Requirements Seminar*, May 1–2, at the Monterey Convention Center and Sheraton Hotel in Monterey, California. The speaker is Bob Braden, a Project Leader at the USC Information Sciences Institute. He co-authored RFC 1009 with Jon Postel, and has been serving as editor of the Host Requirements RFC.

This seminar will provide an introduction to two important documents concerning the implementation of TCP/IP protocols for hosts and gateways: an RFC to be published shortly that is entitled "Requirements for Internet Hosts," and RFC 1009 called "Requirements for Internet Gateways." These RFCs, which contain the latest amendments to the protocol specifications and summarize required, recommended, and optional features, should be invaluable reference documents for implementors of the TCP/IP protocol suite. The host document fills in many of the engineering details left undefined in the original protocol RFCs, and contains common pitfalls as well as implementation suggestions. It covers all the major host protocols.

The seminar will survey these documents and highlight the key issues and requirements. The speaker will discuss and answer questions on the reasoning behind the specifications in the document. For more information, call us at 415-941-3399.

A Network Management Language

by Unni Warrier, Unisys Corporation

Introduction

The size and complexity of computer communication networks have been growing steadily. Along with this growth has come the recognition of network management as an important aspect of computer networking. A goal of network management is to monitor, record, and control the dynamically evolving global network state. Although network management has been largely a manual activity, computer-assisted network management is beginning to produce more automated, intelligent, and responsive tools/techniques for management.

Network management is the responsibility of a staff dedicated to maintaining the services of the network. The network administrator is usually a computer or telecommunications professional who will use computer-based tools to carry out her or his management duties. These tools will be built by a network management application programmer who requires an appropriate language interface with which to program applications. It is the purpose of this article to point out the importance of such a language interface, which we call the *Network Management Language* (NML). This article is based on [1], and the interested reader is referred there for more complete information.

Architecture

The network is assumed to consist of end systems (e.g., user nodes) and intermediate systems (e.g., routers, bridges) that communicate according to a set of common protocols. These systems may be interconnected by local and wide area networks. The network may have one or more *Network Managers* (NMs), which are end systems from which management functions are performed. Thus the NM is a focal point for the collection and processing of management information. The use of multiple NMs generally depends upon the magnitude of the management task.

The NM is thought of as a set of *Application Units* (AUs) that provide high-level interface functions between the (human) network administrator and the network management system. For instance, an AU might be an application that graphically displays the network connectivity map. These AUs include NML commands which are interpreted by the NML interpreter. NML commands are translated by the interpreter into primitive commands understood by the communication system. The interrelation of the AUs, NML interpreter, and communication system is illustrated in Figure 1.

The communication system allows nodes to exchange information by means of specific protocols. In order to communicate management information and commands to managed nodes, the NM uses a *Management Information Exchange Protocol* (MIXP). The MIXP uses, in turn, other protocols to transmit and receive data, e.g. remote operations protocol, presentation protocol, etc. Examples of MIXPs abound, including the *Common Management Information Protocol* (CMIP), the *Simple Network Management Protocol*, (SNMP) the emerging *CMIP over TCP* (CMOT) standard, and numerous other proprietary protocols.

continued on next page

A Network Management Language *(continued)*

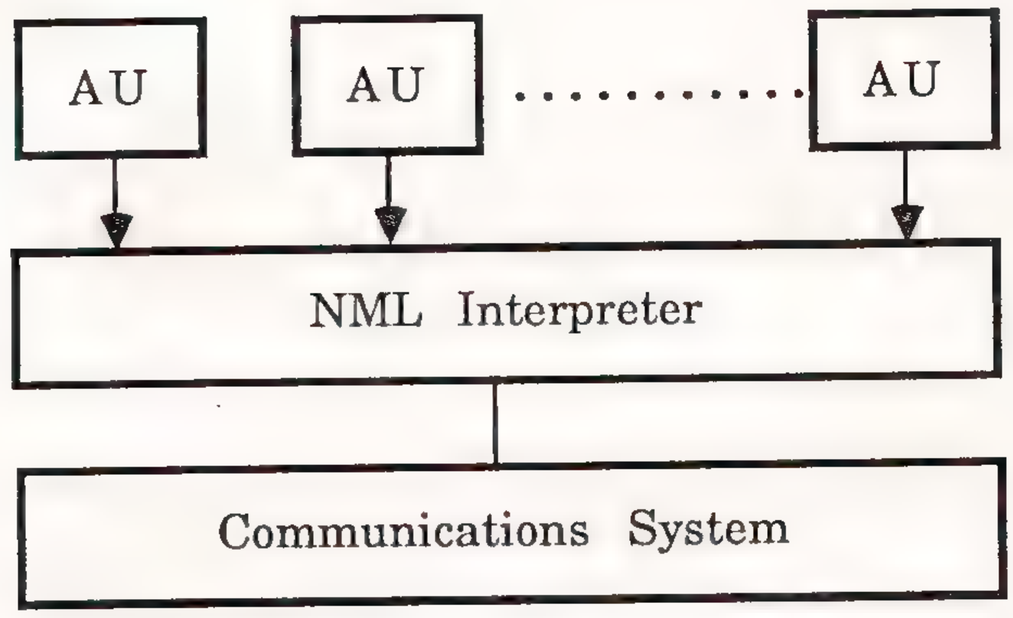


Figure 1: Network Manager

Rationale for NML

Referring to Figure 1, we observe that there is no absolute need for an NML interpreter. AU programs could be written to provide a direct interface to the MIXP (which is part of the communication system). In fact, such a direct interface is desirable in certain situations. A major thesis of this article is that a well-designed network management system will offer the network management application programmer a view of the network that sits at a significantly higher level than the MIXP. It is heartening to note that, in the Internet world, work is now beginning on a canonical Applications Programming Interface (API), which could evolve to become an NML.

Vendors of communication hardware and software will make substantial investment in delivering network management products. Such a product offering entails engineering the network nodes to support management functions. In addition, there will be high recurring and nonrecurring engineering costs associated with building network management applications incorporating sophisticated graphics, database and expert system packages. Vendors will certainly wish to protect their investments in network management applications. We believe that an NML will allow network management applications to be constructed in an economical way.

NML is seen as a canonical interface between the network management application programmer and the MIXP. The following advantages can be derived from the use of NML:

• *Reduce the semantic gap between MIXP and the application programmer:* The MIXP is a low-level communication protocol from the application programmer's point of view. The primitives provided by an MIXP are inappropriate for use by a network management application programmer. Such primitives have a very elementary model of operation, calling for the immediate firing of the underlying state machine, requiring long lists of parameters, and applying only to single entities of the network. Thus it is reasonable to expect an interface with more expressive power for the network management application programmer.

• *Cope with multiple MIXPs:* As noted previously, several MIXPs are currently in use. Vendors may wish to support proprietary protocols and make smooth transitions to internationally recognized standards. Changing AUs whenever a new MIXP is used could be costly.

If one were to build m different AUs on top of n different MIXPs, then $m \times n$ implementations would be required for full connectivity. If there is a single canonical NML, then only n different NML interpreters are necessary. As m is typically large and n is small in a given network, the advantage of NML is immediately apparent. This argument also applies to MIXPs that are not fully mature (like CMIP). By writing applications in a well-specified language like NML, one can insulate oneself from the vagaries and vicissitudes of the standards-making process.

• *Promote standardization:* It may be mutually beneficial for a number of organizations to adopt a common NML interface. The major benefits would be reusability and transportability of application software. This may also encourage the adoption of widely accepted programming conventions for network management applications.

The disadvantage of NML is that it inserts another layer (the NML interpreter) into an already heavily layered system. Preliminary analysis leads us to believe that the performance penalty will not be substantial. We believe that the penalty of using an NML will be offset by the advantages outlined earlier. The decision to employ an NML is clearly a design tradeoff that must be made by the system designer. Two of the objectives of our research is to explore the impact of NML on system performance and to study techniques for providing acceptable levels of performance.

NML requirements and principles of operation

The NML will perform certain high-level operations at the network management application programmer's request. The requirements for NML are based on what we feel are proper abstractions to export to the network management application programmer, given that such a programmer would wish to perform a number of well-known network management applications such as down-line load, network initialization, and the setting and getting of attributes.

The principal need of the Network Manager is to be able to communicate with, and manage, the devices in the network. This requires some means in the NML system to be able to issue management commands which will identify and operate on a network resource or a set of resources.

Another important need is for the Network Manager to access data in the *Management Information Base* (MIB). The MIB is an abstraction that represents the state of the network's many managed objects. The MIB is central to network management, and the network administrator must have capabilities to review the data and perform complex manipulations on it. Thus there should be a way in the NML to be able to issue MIB manipulation commands.

A number of explicit requirements for NML may be derived. In analyzing the requirements for NML we have attempted to make choices that do not restrict the policies that a network management system would wish to enforce.

Our point of departure is to view network management as the manipulation of the MIB where the MIB is understood to be a database that somehow reflects the past, present, and future states of the network.

continued on next page

A Network Management Language *(continued)*

Almost all control and monitoring functions will be then directed through the MIB. The MIB is a logical concept and its design and implementation are outside the scope of this article. NML will include linguistic support for the following requirements.

• *MIB-oriented network management:* The MIB provides a comprehensive state description of the network. Implicit in a MIB oriented paradigm of network management is the assumption that the MIB is a temporal database that incorporates time as a primitive attribute of any record. Most NML mechanisms should therefore be based upon the querying and modifying of the MIB as the fundamental model for monitoring and controlling all network resources.

• *Symbolic naming:* The network administrator will refer to network resources and entities by symbolic names. There will be a need to address groups of resources via a single name. There may also be a need to call objects by nicknames or aliases. Symbolic naming is therefore essential in the NML.

• *Grouping of operands:* Management of a large network entails applying perhaps several distinct operations to each end or intermediate system. It is unreasonable to ask that the network management application programmer perform each of these individual operations. Therefore the NML should enable the network management application programmer to perform groups of operations on groups of devices.

• *Procedural mechanisms:* The ability to specify modules that implement a given function is fundamental to nearly every higher level language. Invoking procedures or macros by symbolic reference aids in the development and sharing of programs. The network management application programmer will certainly benefit from such a facility.

• *Temporal control:* The NML must allow operations to be specified as occurring at specific time points. This requires being able to schedule operations with a great deal of control. The desired mechanisms include deferring operations to later times, specifying repeated operations with a given periodicity, or in general laying out any reasonable schedule for a sequence of operations.

• *Error handling:* Devices frequently fail and data transmission is inherently unreliable. These errors cause problems for network management and must therefore be detected and possibly counteracted. The logging of errors and anomalous conditions should be provided by the NML.

NML definition

Having established the above requirements, it is now possible to define an NML. It must be noted that many different types of NML may be defined, based on the desired abstraction of the global network state. For example, it is possible to view the global state as either relational or as object-oriented. The relational scheme presents the network as a collection of tables. The object-oriented scheme presents the network as a collection of objects and operations on these objects. The advantage of the relational scheme is that existing, well-understood relational technology can be used in the implementations.

The disadvantage is that the modeling of events and actions, which are natural to a network but do not naturally fit into tables, becomes clumsy. This can be solved with the use of the object-oriented scheme, but then the implementor is faced with dealing with the emerging technology of object-oriented databases. Such databases are not yet well understood, and are not commercially available. Rather than deal with emerging technology on two fronts (networks and databases), it was decided to take the conservative approach and define a relational NML. However, it should be noted that the choice of paradigm is irrelevant to the generic language extensions needed, and the type of system architecture proposed below.

The NML interface

One possible representation of the global network state is as a collection of *tables* in a *relational database*. Thus the state variable of the network would be the *conceptual schema*, which would be fairly static over time. The advantage of this representation is that common relational algebra can be used for accessing state variables.

Obviously, this global network state is not static, but changes over time. Hence the network management system has to provide the means of storing the history of network state changes somewhere in the system. It should also provide the means of manipulating this historical data. One method of such organization is by postulating *versions* of the information in the schema, i.e., at the time that a single data item in the database changes, a new version of that item has become available from that particular point of time.

Thus the interface presented to the applications by the NML is a *versioned relational database*. The task of the NML, then, is to accept commands in a versioned relational query language, interpret these commands into ISO-specific MIXP function invocations, collate the results of these invocations, and translate them into relational terms before providing these results to the user process.

The tables that keep the history and current values of the state of the network could exist solely at a single point (a centralized database), only at the end devices (a distributed database), or at both the centralized site and the end devices (a distributed database with centralized replication). In addition, some of the data may exist under the control of *sub-managers*. It must be noted that the NML is independent of the particular choice of data location in the network.

Syntax

Rather than define an entirely new language for network management, it was decided that the NML would be an extension of some existing command language. SQL was chosen, since it has been accepted widely in industry, and has been standardized by the American National Standards Institute (ANSI). In addition, SQL is also a Military Standard. Thus the network management database would be thought of as an SQL standard database. The NML imports all the features of SQL. In particular, the Schema Definition Language and the Module Language are identical to SQL. However, we note that user needs for creation of schemas, redefinitions of schemas, creation of modules, can be met via these languages. The Data Manipulation Language (DML) is extended to include a network management specific concept of time (or versions). Here we concentrate on the NML extensions to the SQL DML, and will assume regular SQL functions to be given.

continued on next page

A Network Management Language *(continued)*

It was decided that, in the network management context, three types of DML extensions made sense in accessing versions of the SQL database. The AT clause supports the notion of data having a specific time context (roughly, from a specific version of the data items). The FOR clause supports the need for periodic access of data over a certain duration (i.e., from consequent versions of the same data items). The EVERY clause supports the need for a specific periodicity for the FOR clause.

NML statements

The syntax for the NML given here is an extension of the SQL syntax. The generic NML statement is:

```
<NML statement> ::= FORCE <SQL statement>[AT <time-value>]
[FOR <for-clause>]
```

Embedded NML

In parallel with *embedded SQL*, we can define an embedded NML that will let NML be invoked from standard programming languages. Although embedded SQL is not strictly part of the ANSI standard, the Annexes to the standard give a complete treatment of embedded SQL for some standard programming languages. The translation of this treatment to NML is obvious. The paper [1] provides a fuller definition of the NML syntax.

Semantics

Since NML commands are extensions of the commands from SQL, the semantics with respect to the <SQL statement> part of the syntax are just the SQL semantics, and will not be discussed here. What will be discussed are the meanings of the various extensions to SQL in the NML DML.

Immediate NML statements

These commands let the user manipulate instances of the data in the database via extended SQL commands. The format is:

```
<NML statement> ::= <SQL statement> [AT <time-value>]
 FOR <duration> [EVERY <period>]
```

The AT and FOR clauses are interpreted as a list of all the versions of the data in the database participating in the <SQL statement>.

Forced NML commands

The forced commands are a concession to the fact that network management may involve replication in the data organization, with inconsistency between the replicated copies. For example, the organization may involve replication of all distributed tables (at the devices) at a central site, with the update frequency at the central site being slow enough to cause some inconsistency with the tables at the distributed sites. The FORCE command, then, makes the NML system seek the values at the distributed sites, rather than at local centers of replication.

NML system architecture

The NML will have a supporting architecture that will provide the necessary translations to permit the NML to reside on top of any MIXP. Such an NML system will translate NML statements into MIXP management commands, and retranslate the results of these commands to NML semantics as responses. Figure 2 shows the relationship of the NML architecture and its components to the MIXP. The NMLS will generally be present at manager nodes only. If the system allows the partitioning of the management functions between multiple managers, or hierarchical managers, more than one system may behave as manager as well as agent.

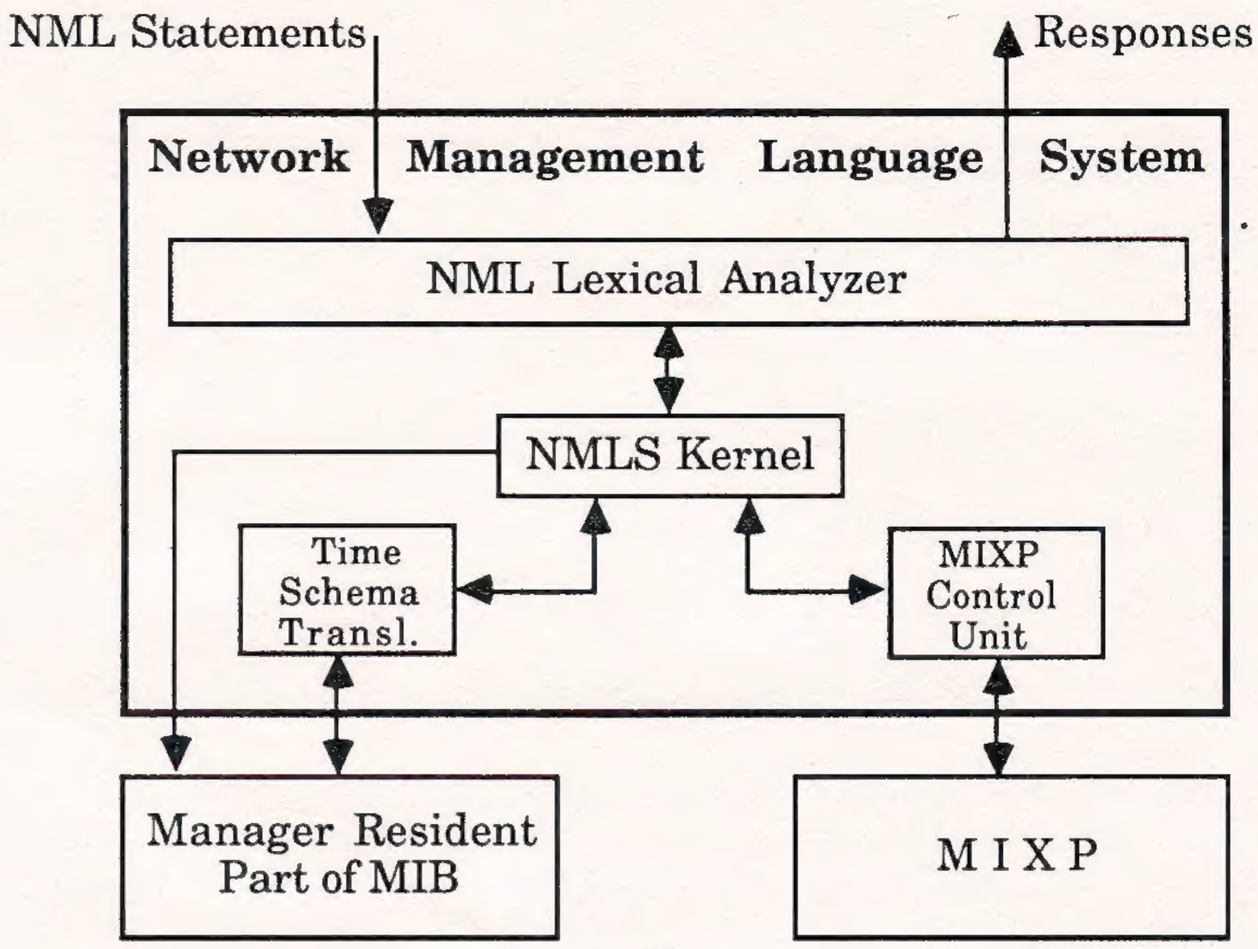


Figure 2: The Network Management Language System

This type of system would be responsible for management of some "pure agents," and simultaneously be accessible for queries by other managers. NMLS would be present in this type of a sub-manager.

In the OSI (or CMOT) context, NMLS is a user of CMIP and Manager MIB services at the application layer, and a provider of a high level command service to network management applications. It is entirely feasible that other applications that provide interfaces to alternative MIXPs will coexist in the NMLS. Similarly, the issue of Proxy Management, whereby a Proxy Manager acts as an agent for a collection of nodes that do not talk CMIP/CMOT, is also possible with our architecture of NML. For example, a Proxy Manager can be constructed from an implementation of NML with two MIXP interfaces, one interfacing to CMIP and the other to SNMP. Such a manager will be able to manage SNMP devices on behalf of another manager that has only a CMIP interface.

Conclusion

This article has introduced the concept of the NML, which provides the network management application programmer with powerful constructs for building network management applications. The NML provides its user with an integrated view of all network resources by allowing the programmer to specify queries on the MIB and propagating operations on the MIB to corresponding physical entities in the network. We have presented arguments in favor of NML based on its expressiveness, potential cost savings, and versatility in adapting to changes in underlying communication services. Finally, we have defined an NML that is an extension of the SQL standard, resulting in a relationally complete language.

References

[1] Warrier, U., Relan, A., Berry, O., Bannister, J., "A Network Management Language for OSI Networks," Proc. ACM SIGCOMM 88.

UNNIKRISHNAN WARRIER is currently working as a Research Scientist with the Unisys Corporation. He has been working in computer science for the last ten years, concentrating on network architecture, modeling and management. He has published several papers in these areas. He is a member of the IEEE Committee on Networks Operations and Management, IETF NetMan working group, the ANSI X3T5.4 committee on network management, and the NIST Network Management Special Interest Group.

CONNEXIONS
480 San Antonio Road
Suite 100
Mountain View, CA 94040

Bulk Rate
U.S. POSTAGE
P A I D
SAN JOSE, CA
PERMIT NO. 1